The Dissertation Committee for Dirk Thiele
certifies that this is the approved version of the following dissertation:

Novel Methods that Improve Feedback Performance of Model Predictive Control with Model Mismatch

Committee:

Robert H. Flake, Co-Supervisor

Thomas F. Edgar, Co-Supervisor

Joydeep Ghosh

Ari Arapostathis

Glenn Masada

Wilhelm K. Wojsznis

Novel Methods that Improve Feedback Performance of Model Predictive Control with Model Mismatch

by

Dirk Thiele, Dipl.-Ing. (FH), B.S. EE

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May, 2009

Dedicated to my Family, Claudia, Sonja and Lara Thiele

Acknowledgements

I would like to thank all of the people who helped me complete this challenging journey. The completion of my dissertation would not have been possible without the encouragement and support of my family, especially my wife, and children. This accomplishment would also not have been possible without my wonderfully caring parents and the great teachers I have had along the way, from elementary through high school, and at the university level. They gave me the skills, confidence and inspiration that have enabled me to keep pursuing my goals. It has been a great honor to learn and research with my two advisors Prof. Thomas Edgar and Prof. Robert Flake. Their outstanding teaching skills and extraordinary subject matter expertise have made the difference for me and my research. Prof. Joe Qin contributed greatly to my research by continually supplying challenges and inspiration. I would like to thank the outstanding Professors Joydeep Ghosh, Ari Arapostathis and Glenn Masada for also serving on my thesis committee.

During the last decade I had the fortune to be mentored by many talented engineers with brilliant personalities such as Terry Blevins, Dennis Stevenson, Willy Wojsznis, Greg McMillan, Mark Nixon, John Gudaz, Ken Beoughter, Mike Apel, Christian Diedrich, Todd Maras and Noel Bell. Together we were able to solve very tough industrial challenges, invent new technologies and build wonderful friendships.

Lastly I would like to thank the many exceptional graduate students and visiting scholars, whom I had the pleasure to study with, including: Jürgen Hahn, Sébastien Lextrait, Kevin Chamness, Lina Rueda, Koichi Onodera, David Castineira, Xiaoliang Zhuang, Yang Zhang, Claire Schoene, Elaine Hale, Terry Farmer, Ivan Castillo, Hyung Lee, Amogh Prabhu, Dan Weber, Scott Harrison, Chris Harrison and John Hedengren.

Novel Methods that Improve Feedback Performance of Model Predictive Control with Model Mismatch

Publication No._____

Dirk Thiele, Ph.D.

The University of Texas at Austin, 2009

Supervisors: Robert H. Flake, Thomas F. Edgar

Model predictive control (MPC) has gained great acceptance in the industry since it was developed and first applied about 25 years ago [1]. It has established its place mainly in the advanced control community. Traditionally, MPC configurations are developed and commissioned by control experts. MPC implementations have usually been only worthwhile to apply on processes that promise large profit increase in return for the large cost of implementation. Thus the scale of MPC applications in terms of

number of inputs and outputs has usually been large. This is the main reason why MPC has not made its way into low-level loop control.

In recent years, academia and control system vendors have made efforts to broaden the range of MPC applications. Single loop MPC and multiple PID strategy replacements for processes that are difficult to control with PID controllers have become available and easier to implement. Such processes include deadtime-dominant processes, override strategies, decoupling networks, and more. MPC controllers generally have more "knobs" that can be adjusted to gain optimum performance than PID. To solve this problem, general PID replacement MPC controllers have been suggested. Such controllers include forward modeling controller (FMC)[2], constraint LQ control[3] and adaptive controllers like ADCO[4]. These controllers are meant to combine the benefits of predictive control performance and the convenience of only few (more or less intuitive) tuning parameters. However, up until today, MPC controllers generally have only succeeded in industrial environments where PID control was performing poorly or was too difficult to implement or maintain. Many papers and field reports [5] from control experts show that PID control still performs better for a significant number of processes. This is on top of the fact that PID controllers are cheaper and faster to deploy than MPC controllers. Consequently, MPC controllers have actually replaced only a small fraction of PID controllers.

This research shows that deficiencies in the feedback control capabilities of MPC controllers are one reason for the performance gap between PID and MPC. By adopting knowledge from PID and other proven feedback control algorithms, such as statistical process control (SPC) and Fuzzy logic, this research aims to find algorithms that demonstrate better feedback control performance than methods commonly used today in model predictive controllers. Initially, the research focused on single input single output

(SISO) processes. It is important to ensure that the new feedback control strategy is implemented in a way that does not degrade the control functionality that makes MPC superior to PID in multiple input multiple output (MIMO) processes.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Process control is a major consideration of almost all installations of chemical, pharmaceutical and refining industries, and a multibillion-dollar business worldwide. Although the best possible control has not always been a major focus in industry over the past 50 years, in recent years new plants are increasingly designed with controllability and optimizability in mind. Additionally, many existing plants are being renovated with this objective as well. Optimization in this sense includes geometry of the installed hardware, such as reactors, tanks, pipes location, etc., but also the locations of control elements and measurements. With the increasing cost of natural resources and the effective costs associated with undesirable emissions, energy consumption has also become a great factor in plant design. Control performance monitoring in combination with controller retuning or model scheduling can dramatically improve the efficiency of industrial plants, thereby saving millions of dollars annually. Another technique that has become increasingly popular among industrial plants in recent years is abnormal situation monitoring and prevention. Modern device and control system include novel sensor designs and embedded statistical algorithms that are able to predict potential failures or upcoming maintenance cycles. Common measurement device and actuator failures include plugged lines, heat transfer change due to fouling, and control element failures or nonlinearities due to build-up in valves or failures in positioners. Predictive maintenance systems can dramatically increase the uptime of plant operations and prevent costly and dangerous manifestations of unexpected

shutdowns. The reliability of these techniques has significantly increased in the last decade.

1.1 Process model mismatch in industrial processes

In the United States and Canada, process control systems are usually commissioned by a local business partner of the control system vendor. However, some end users have enough in-house capacity to commission and maintain a complete control system themselves. In Europe and Asia, often a local subsidiary representing the control system vendor is employed for that task. In either case, the commissioning costs of a control system are substantial, and it is rarely practical to pay detailed attention to every control loop configuration in a plant. About 90% of all loops are controlled by traditional linear controllers such as PID. Most instances of advanced control strategies are also linear, such as model predictive control (MPC). While linear controllers are predominantly used, the majority of real processes exhibit nonlinear behavior. The obvious consequence of this discrepancy is that model mismatch is unavoidable. Unaddressed model mismatch not only results in suboptimal control performance, but also nullifies many of the advantages of the aforementioned technologies that may have otherwise improved control performance and uptime. Therefore, model mismatch is not only costly, but also diminishes cost savings of other technologies. This is the main motivation for this research.

Practical ways to address model mismatch resulting from nonlinearity include linearization of control elements and transmitters, and controller gain scheduling. Simple linearization functions are built into most automation systems and intelligent field devices. One practical method that is frequently used for addressing model

mismatch from nonlinearity and plant drift is tuning a controller for the worst case scenario (e.g., highest process gain) and accepting suboptimal tuning for other regions of the process. The default tuning parameters of an industrial PID controller are conservative in order to work, at least initially, for a variety of processes applications. Unfortunately, one finds that frequently controller tuning parameters are left at their default values indefinitely.

Model mismatch that results from identification error or from plant drift is more difficult to address—it is hard to detect because sufficient process perturbation is required, which contradicts the objective of control. Furthermore, it is hard to distinguish from unmeasured disturbances. Chapter 2 will go on to discuss previous research in this area.

1.2 Model Predictive Control

Linear model predictive control refers to a class of control algorithms that compute a manipulated variable profile by utilizing a linear process model to optimize a linear or quadratic open-loop performance objective subject to linear constraints over a future time horizon. The first move of this open loop optimal manipulated variable profile is then implemented. This procedure is repeated at each control interval, and the process measurements are used to update the optimization problem as shown in Figure 1.1.

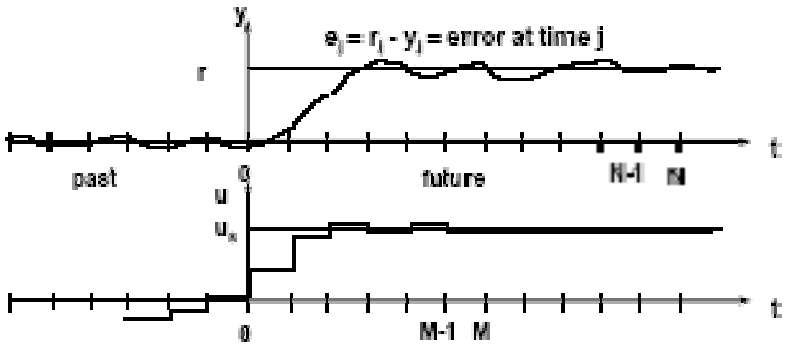

Figure 1.1: receding horizon control: m control moves are calculated at every execution period, only the first one u0 is implemented

This class of control algorithms—also referred to as receding horizon control or moving horizon control—has several advantages for application in chemical process control. The controller uses a linear transfer function, state space, or convolution plant model. These models can be obtained from process tests using time series analysis techniques that do not require a significant fundamental modeling effort. Multivariable processes can easily be handled by superposition of the linear models. Optimization of the open-loop performance objective is performed by either linear or quadratic programming algorithms.

Shown below is the discrete dynamical system model used by the controller is the state-space formulation, in which y is the vector of outputs, u is the vector of inputs, and x is the vector of states.

$$\hat{x}_{k+1} = Ax_k + Bu_k \quad k = 0,1,2,... \tag{1.1}$$

$$\hat{y}_k = C\hat{x}_k \tag{1.2}$$

where y_k is the process output and u_k is the controller output. This receding horizon regulator is based on the minimization of the following infinite horizon open-loop quadratic objective function at time k,

$$\min_{u^N} \sum_{j=0}^{\infty} (y_{k+j}^T Q y_{k+j} + u_{k+j}^T R u_{k+j} + \Delta u_{k+j}^T S \Delta u_{k+j}) \tag{1.3}$$

where Q is a symmetric positive semidefinite penalty matrix on the outputs with y_{k+j} computed from Equation (1.1). R is a symmetric positive definite penalty matrix on the inputs in which u_{k+j} is the input vector at time j in the open-loop objective function. S is a symmetric positive semi-definite penalty matrix on the rate of change of the inputs in which $\Delta u_{k+j} = u_{k+j} - u_{k+j-1}$ is the change in the input vector at time j. The vector u^N contains the N future open-loop control moves as shown below.

$$u^N = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix} \tag{1.4}$$

At time k+N, the input vector u_{k+j} is set to zero and kept at this value for all $j \geq N$ in the open-loop objective function value calculation.

The receding horizon regulator computes the vector u^N that optimizes the open-loop objective function in Equation (1.3). The first input value in u^N, u_k, is then injected into the plant. This procedure is repeated at each successive control interval with feedback incorporated by using the plant measurements to update the state vector at time k.

The infinite horizon open-loop objective function in Equation (1.3) can be expressed as the finite horizon open-loop objective shown below.

$$\min_{u^N} \Phi_k = y_{k+N}^T \overline{Q} y_{k+N} + \Delta u_{k+N}^T S \Delta u_{k+N} + \min_{u^N} \sum_{j=0}^{N-1} (y_{k+j}^T Q y_{k+j} + \Delta u_{k+j}^T S \Delta u_{k+j}) \quad (1.5)$$

The output penalty term in Equation (1.3) has been replaced with the corresponding state penalty term in Equation (1.5). Determination of the terminal state penalty matrix, $\overline{Q}$, depends on the stability of the plant model.

For stable systems, $\overline{Q}$ in Equation (1.5) is defined as the infinite sum in Equation (1.6).

$$\overline{Q} = \sum_{i=0}^{\infty} (A^{T^i} C^T Q C A^i) \quad (1.6)$$

This infinite sum can be determined from the solution of the following discrete Lyapunov equation:

$$\overline{Q} = C^T Q C + A^T \overline{Q} A \quad (1.7)$$

There are standard methods available for the solution of this equation. Straightforward algebraic manipulation of the quadratic objective presented in Equation 4 results in the following quadratic program for u^N:

$$\min_{u^N} \Phi_k = (u^N)^T H u^N + 2(u^N)^T (G x_k - F u_{k-1}) \quad (1.8)$$

The matrices H, G, and F are computed as shown below with determined from Equation (1.7).

$$H = \begin{bmatrix} B^T \overline{Q} B + R + 2S & B^T A^T \overline{Q} B - S & \dots & B^T A^{T^{N-1}} \overline{Q} B \\ B^T \overline{Q} A B - S & B^T \overline{Q} B + R + 2S & \dots & B^T A^{T^{N-2}} \overline{Q} B \\ \vdots & \vdots & \ddots & \vdots \\ B^T \overline{Q} A^{N-1} B & B^T \overline{Q} A^{N-2} B & \dots & B^T \overline{Q} B + R + 2S \end{bmatrix}$$

$$G = \begin{bmatrix} B^T \overline{Q} A \\ B^T \overline{Q} A^2 \\ \vdots \\ B^T \overline{Q} A^N \end{bmatrix}, \; F = \begin{bmatrix} S \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

The discussion of instable systems begins with partitioning the Jordan form of the A matrix into stable and unstable parts in which the unstable eigenvalues of A are contained in J_u.

$$A = VJV^{-1} = \begin{bmatrix} V_u & V_s \end{bmatrix} \begin{bmatrix} J_u & 0 \\ 0 & J_s \end{bmatrix} \begin{bmatrix} \widetilde{V}_u \\ \widetilde{V}_s \end{bmatrix} \tag{1.9}$$

The stable and unstable modes, z_s and z_u respectively, then satisfy the following relationships:

$$\begin{bmatrix} z^u \\ z^s \end{bmatrix} = \begin{bmatrix} \widetilde{V}_u \\ \widetilde{V}_s \end{bmatrix} x \tag{1.10}$$

$$\begin{bmatrix} z^u_{k+1} \\ z^s_{k+1} \end{bmatrix} = \begin{bmatrix} J_u & 0 \\ 0 & J_s \end{bmatrix} \begin{bmatrix} z^u_k \\ z^s_k \end{bmatrix} + \begin{bmatrix} \widetilde{V}_u \\ \widetilde{V}_s \end{bmatrix} B u_k \tag{1.11}$$

For unstable plants, the finite horizon open-loop objective function in Eq. 4 is subject to the following equality constraint on the unstable modes at time k+N.

$$z^u_{k+N} = \widetilde{V}_u x_{k+N} = 0 \tag{1.12}$$

This equality constraint is required if the unstable modes are not brought to zero at time k+N, they evolve uncontrolled after this time and do not converge to zero. Therefore, the optimal solution to (1.5) must be a vector u^N that zeroes the unstable modes at time k+N.

With the equality constraint ensuring that only the stable modes contribute to the value of Φ_k after time k+N-1, Q for unstable systems can be computed from the stable modes in a manner similar to Equation (1.6).

Muske and Rawlings (1992) show that this regulator formulation guarantees nominal stability for all choices of tuning parameters satisfying the conditions outlined in the previous sections. Nominal stability comes from the evaluation of the state penalty on an infinite horizon even though there are a finite number of decision variables. Previous model predictive controller formulations are finite horizon. The absence of nominal stability in these implementations is a direct consequence of the finite horizon formulation of the control algorithm. Bitmead et al. (1990) demonstrate that nominal stability cannot be guaranteed for a finite receding horizon regulator.

Kwon and Pearson (1978) propose a nominally stabilizing receding horizon regulator based on a finite horizon objective subject to a terminal state constraint. The terminal constraint forces all of the modes of the system to be zero at the end of the horizon instead of only the unstable modes. This constraint leads to aggressive control action with small values of N for both stable and unstable systems since the regulator approaches a deadbeat controller. Feasibility of this terminal constraint also requires that the system be completely controllable. Additionally, Rawlings and Muske's proof requires the systems to be stabilizable and uncontrollable modes to reach origin in infinite steps.

Industrial implementations began with model algorithmic control (MAC) developed by Richalet et al. (1978) and dynamic matrix control (DMC) developed by Cutler and Ramaker (1980). These Shell Oil engineers developed their own independent MPC technology in the early 1970's, with an initial application in 1973.

Cutler and Ramaker presented details of an unconstrained multivariable control algorithm which they named Dynamic Matrix Control (DMC) at the 1979 National AIChE meeting and at the 1980 Joint Automatic Control Conference. In a companion paper at the 1980 meeting, Prett and Gillette described an application of DMC technology to an FCCU reactor/regenerator. DMC technology uses linear step response or impulse response models of the process. The optimal control path is pre-calculated off-line and stored in a large matrix. This controller matrix is then used to compute the on-line moves of the manipulated variables by superposition. As a result, computational cost is reduced drastically in comparison to MPC methods that solve the above optimal equations on-line. Another advantage of DMC technology is that the state variable is calculated intuitively from the model, and represents the explicit future output prediction. This means that the future prediction of process outputs such as constraints is readily available and can be displayed to the user.

Other early technology implementation include IDCOM (1978) by Richalet et al. and linear dynamic matrix control (LDMC), which uses a linear objective function and incorporates constraints explicitly, by Morshedi et al. (1985). Garcia and Morshedi (1986) discuss quadratic dynamic matrix control (QDMC), which is an extension of DMC incorporating a quadratic performance function and explicit in corporation of constraints. Grosdidier et al. (1988) present IDCOM-M, which is an extension of IDCOM using a quadratic programming algorithm to replace the iterative solution technique of the original implementation. Marquis and Broustail (1988) discuss Shell multivariable optimizing control (SMOC), which is a state-space implementation.

1.3 Control performance criteria

The performance of industrial controllers can be measured in various ways. A comprehensive overview of control performance assessment technologies is given in [6]. Different processes may have greatly different quality and safety requirements; thus, plant engineers may use one or multiple performance criteria, such as overshoot, arrest time (time to reach setpoint for integrating processes), oscillation characteristics, integrated error, and integrated absolute error for their evaluation of loop performance. Furthermore, the measured control performance for a given controller may be a tradeoff between setpoint tracking and disturbance rejection behavior. Traditional PID control, which still is the most popular controller choice in the process industries, suffers from this problem. Many practical approaches such as structure modifications or setpoint filtering have been developed to address this problem. Also, modern model predictive control algorithms have been specifically reformulated to eliminate this disadvantage. An example by Badgwell and Muske is described in [20]. Additionally, section 1.4 further discusses tuning techniques for specific control objectives. Finally, a general overview of commercially available MPC algorithms is given by Qin and Badgwell [32].

Shinskey [8] suggests a performance criteria that is a normalized value of the integrated error defined as:

$$\frac{IE}{\Delta q K_q \tau_d} \tag{1.13}$$

where Δq is the unmeasured disturbance, K_q is the process gain, τ_d is the deadtime and IE is the integrated error calculated by:

$$IE = \int_0^\infty (y(t) - SP(t))dt \tag{1.14}$$

where y(t) is the controlled process output variable and SP(t) is the operator setpoint. Since fractional deadtime is a very important factor for control performance, Shinskey shows why controllers that consider deadtime in their equations, such as model-based, direct synthesis and PIDτ_d [8] generally have better control performance in the deadtime dominant region. However, he also shows that in the other regions the control performance of model-based controllers is generally suboptimal, i.e. the normalized integrated error is equal to one throughout [8], which can be seen in Figure 1.2 (internal model control is indicated as IMC). The dashed curve labeled "best" describes the minimum IE attainable for any feedback controller. All controllers are tuned to minimize the integrated absolute error (IAE) in the controlled variable following a step change in the load introduced at the controller output. The IE represented by the flat line at 1.0 on the normalized scale is produced by an unfiltered internal model controller (IMC). This controller shows the lowest performance of all controllers for lag-dominant processes. Additional filtering only increases IE, in direct ratio of the filter time to the deadtime proportion in the loop. A Smith predictor and any other model-predictive controller whose parameters are simply matched to the process parameters show the same performance as IMC.

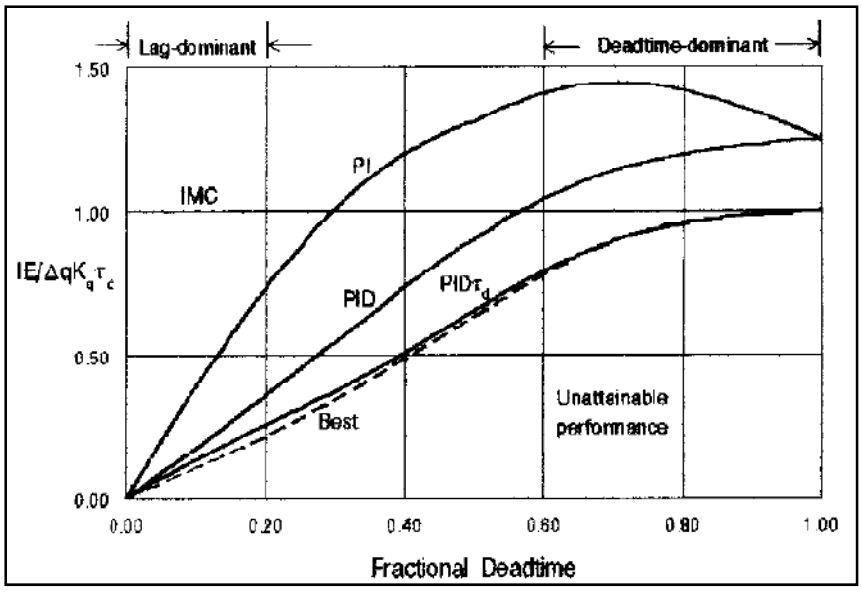

Figure 1.2: Control performance dependence on fractional deadtime for different controllers, Source: F. G. Shinskey [8]

This research does not explore performance improvements for deadtime-dominant processes, because it is generally known that model predictive control is already superior in that region of fractional deadtime. Instead, the research is seeking to improve the control performance in the lag-dominant and intermediate regions. Furthermore, since some tuning settings (that will be discussed in the following sections) can lead to oscillatory behavior around the setpoint, thereby creating negative error, this research uses the normalized value of the *absolute* value of the integral error (IAE) for all comparisons and diagrams:

$$\frac{IAE}{\Delta q K_q \tau_d} \tag{1.15}$$

where Δq is the unmeasured disturbance, K_q is the process gain, τ_d is the deadtime and IAE is the integrated absolute error calculated by:

$$IAE = \int_0^\infty |y(t) - SP(t)|dt \tag{1.16}$$

where y(t) is the controlled process output variable and SP(t) is the operator setpoint.

1.4 Controller tuning and objectives

In general, model-based tuning provides for a trade-off between SP tracking and load rejection performance. Long time constants (i.e., lag dominant processes) are known to cause poor disturbance rejection performance on PID controllers that are tuned for SP tracking performance (Shinskey[24]). This tradeoff on PID controllers can be explained by the fact that a PID controller that is ideally tuned for load disturbance rejection must have a relatively high integral action (a relatively small T_I). Such high integral action is detrimental to the controllers SP change performance. During a SP change the error e remains large for a period of time even while y is approaching SP. With very large K_I, the integral term builds up fast and more than necessary, thus causing SP to overshoot. Consequently, PID tuning targeted for SP change performance has smaller integral action and worse load change performance. Although the process described in the previous section is lag time dominant ($\frac{\theta}{\tau} = 0.1$), the PID controller achieves good SP change and load rejection performance. MPC and PID simulation were comparable in terms of SP tracking, but MPC is much worse than PID for load changes.

A predictive controller should be able to perform similarly for SP changes and load changes because the integral part of a MPC does not suffer the same trade-off as observed for PID. The integral action is merely dependent on $\overline{\overline{K}}$ as shown in the

previous section. Also, the error e does not increase while y is approaching SP in a predictive controller. Theoretically, it can be zero after the first execution cycle. This is only true when assuming that the prediction (state) is updated effectively and doesn't add additional significant time constants. But the fact that the MPC load disturbance performance is slower during the simulation indicates that the state is updated too slowly (i.e., without consideration of the error derivative). In the absence of model mismatch, state update only comes into play during load changes, not during SP changes. Therefore, this simulation is a good preliminary test for the research.

Maciejowski[49] shows how sensitivity functions can be used to analyze a system's sensitivity to unmeasured load disturbances. This analysis differentiates between plant noise *d* and measurement noise *n*.

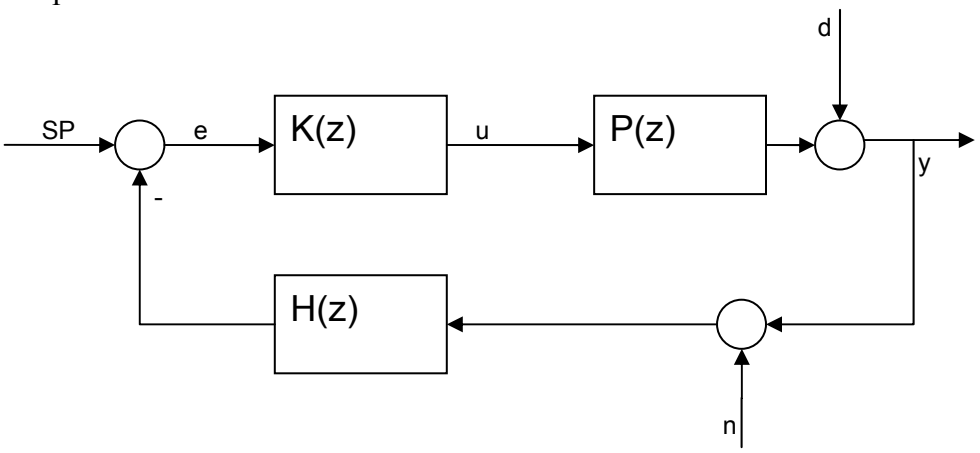

Figure 1.3: Two-degree-of-freedom feedback system

He shows that if a two-degree-of-freedom feedback system as shown in Figure 1.3 is given by equation:

$$y(z)=S(z)\,d(z) + S(z)\,P(z)\,K(z)\,F(z)\,s(z) - T(z) \qquad (1.17)$$

the sensitivity function *S(z)* is:

$$S(z) = [I + P(z)\,K(z)\,H(z)]^{-1} \tag{1.18}$$

and the complementary sensitivity function *T(z)* is:

$$T(z) = [I + P(z)\,K(z)\,H(z)]^{-1} - P(z)\,K(z)\,H(z) \tag{1.19}$$

Systems with 'smaller' sensitivity functions must have better feedback control action. Unfortunately, it is not possible to design a two-degrees of freedom controller that has very low sensitivity and very low complementary sensitivity because these traits are determined by each other:

$$S(z) + T(z) = I \tag{1.20}$$

Since *S(z)* and *T(z)* are complex however it is possible to for them to be large at the same time, i.e., it is possible to design a very bad feedback controller.

From the above equations it can be seen that the setpoint change response can be designed independently from the load-rejection response. However, these two responses will be significantly different, the setpoint response transfer function is *S(z)P(z)K(z),* while the load rejection transfer function is *S(z)*. This fact also supports the findings of the previous section.

The ideal continuous time domain PID controller is expressed in the Laplace domain as follows:

$$U(s) = G_c(s)E(s) \tag{1.21}$$

where

$$G_c(s) = K_c(1 + \frac{1}{T_i s} + T_d s) \tag{1.22}$$

and where K_c is the proportional Gain, T_i is the integral time constant and T_d is the derivative time constant. Tuning PID controllers always presents the challenge of correctly specifying the tradeoff setpoint tracking vs. disturbance rejection performance. A specific textbook tuning method will favor either setpoint tracking or disturbance rejection. Many model-based tuning rules match the internal parameters of PID to internal parameters of a model for the process to be controlled. Methods such as pole cancelation and lambda tuning match the integral time of the controller to the dominant time constant of the process. The controller gain K is set to achieve a certain closed-loop time constant and a certain setpoint change response (e.g., no overshoot). Since the resulting integral action of such controllers is relatively small, it exhibits very good setpoint change performance, but poor disturbance rejection performance. Empirical PID tuning methods, such as Ziegler-Nichols, are specifically designed for disturbance rejection performance. The integral action of such controllers is strong enough to return the process variable to the setpoint very quickly, but leads to undesired setpoint overshoot. Figure 1.4 compares the load change and setpoint change responses of two PID controllers. Both control the same process with the following transfer function: $G(s) = \frac{1}{50s+1} e^{-s}$. The tuning for the first PID controller is obtained by using IMC tuning rules $\tau_{i_{IMC}} = \tau_1 + \frac{\theta}{2}$ resulting in K_C=20.2, T_i=50.5s, T_d=0.49505s, which favors setpoint tracking performance. The tuning for the second PID controller is obtained by using Skogestad's tuning rules [9] $\tau_{i_{Skogestad}} = 8\theta + \tau_2$ resulting in K_C=25, T_i=8s, T_d=0s, which is significantly different and favors load disturbance rejection

performance. It can be seen that the PID with more integral action performs much faster if the load disturbance is introduced, while the PID with less integral action performs much better during setpoint change because it has no overshoot.

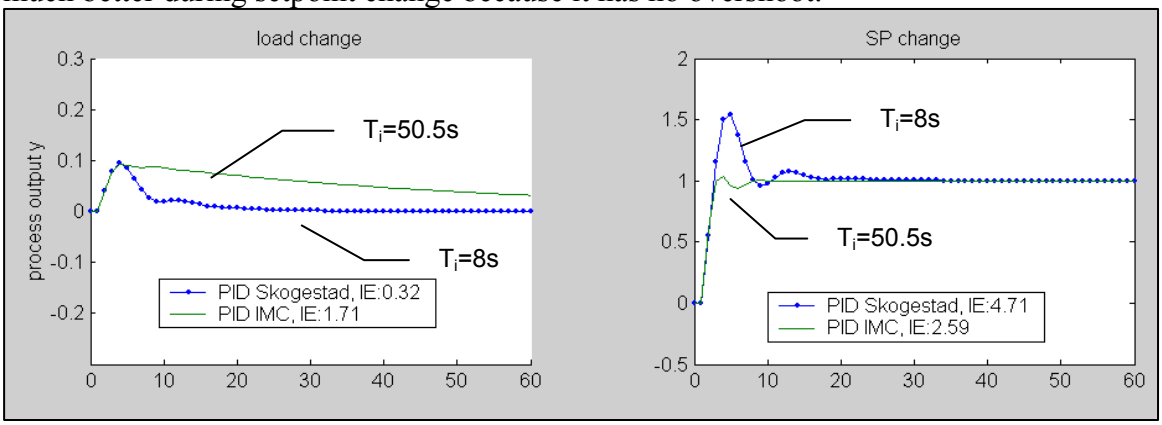

Figure 1.4: Tradeoff between setpoint tracking and load rejection performance caused by different tuning rules that result in significantly different integral time constants. left: IMC (K_c=20.2 ***T_i=50.5s*** T_d=0.49505s); right: Skogestad (K_c=25 ***T_i=8s*** T_d=0) tuning, process model parameters: G=1, τ_1=50s, Θ=1s

In rare occasions the purpose of a loop is only disturbance rejection (e.g., buffer tank level with no setpoint changes) or only setpoint tracking (e.g., secondary loop in cascade strategy with no disturbances). In those cases it may be easy to choose a tuning rule. However, in many plants the aforementioned tradeoff is frequently overlooked entirely and a default tuning method is chosen. Industrial users of PID control have developed numerous methods to overcome this limitation of PID tuning. Such methods include setpoint filtering and two-degree-of-freedom structures [10]. In both of these methods the tuning rules favor disturbance rejection performance and the controller reaction to setpoint changes is artificially reduced. If setpoint filtering is chosen, setpoint changes by the operator are filtered with a simple filter, as shown in Equation

(1.24). This effectively flattens the step change of the operator and creates a smaller slope as setpoint, thereby preventing overshoot.

$$\frac{SP*}{SP} = \frac{1}{\tau_f s + 1} \tag{1.23}$$

Where SP is the operator and SP* is the working setpoint. Equation (1.24) shows a two-degree-of-freedom approach that factors the controller gain out, effectively separating the setpoint and disturbance behavior [10]. The setpoint weighting factor β is bounded, $0 < \beta < 1$, and becomes a new tuning parameter for the controller.

$$p(t) = \bar{p} + K_c\left[\beta SP(t) - y(t)\right] + K_c\left[\frac{1}{\tau_I}\int_0^t e(t)dt - \tau_D\frac{dy_m}{dt}\right] \tag{1.24}$$

In the case of lambda tuning for integrating processes, overshoot can be prevented by setting the filter factor equal to lambda [7]. PID algorithms are often referred to as two-degree-of-freedom (also known as 2DOF) if the calculation of derivative action is factored out of the PID equation and only applied to the process variable (instead of applying it to the error which reflects setpoint changes). Shinskey proposed a modification [8] which eliminates the performance tradeoff and improves performance for deadtime-dominant processes (See also Figure 1.2). In general, different tuning methods have to be chosen for different control objectives. The performance tradeoff described above is one of the reasons why so many tuning methods have been proposed for PID tuning, along with the fact that tuning rules use different input variables. While the tuning examples shown in Figure 1.4 calculate tuning parameters based on a process model, other methods calculate it based on other process characteristics. For example, Ziegler-Nichols rules ask for critical gain and

critical frequency, which may be easy to determine for some mechanical processes but not practical for many industrial chemical processes. Aström and Hägglund developed a relay oscillation method that can bring a system into sustained self-oscillation at controllable amplitude, resulting in equally useful oscillations that are much less intrusive to the process [42]. In combination with original or modified Ziegler-Nichols tuning rules, this technique is used in many commercial products, such as the digital automation system DeltaV [43]. O'Dwyer summarizes about 300 different tuning methods in his research [19].

Model predictive controllers generally do not have a performance tradeoff between setpoint tracking and disturbance rejection because the terms for error and move penalty are inherently separate.

$$\hat{x}_{k+1} = Ax_k + Bu_k \quad k = 0,1,2,... \tag{1.1}$$

$$\hat{y}_k = C\hat{x}_k \tag{1.2}$$

$$\min_{u^N} \sum_{j=0}^{\infty} (y^T_{k+j} Q y_{k+j} + u^T_{k+j} R u_{k+j} + \Delta u^T_{k+j} S \Delta u_{k+j}) \tag{1.3}$$

where Q, R, S are the penalty weights for error, controller move and incremental move, x_k is the model state matrix, y_k is the process output and u_k is the controller output.

However, they still need to be tuned for a specific multivariable process control objective. While the process model is always matched with the internal structure of an MPC controller (e.g., state space with state space MPC formulation), additional tuning parameters determine the behavior with respect to setpoint change and disturbance rejection. The penalty vectors can be used to emphasize one variable over others in

accordance with the control objective for the specific process as defined by the end user. If model mismatch is suspected, Q and R can also be used to make the controller more robust (i.e., detuning). However, methods such as funnel control or reference trajectory have a more obvious impact on robustness as they effectively filter the error vector—which is why they are the preferred means for engineers and operators to tune model predictive controllers in industrial process applications. Since a model predictive controller inherently "matches" the process, the control moves are always optimal for the specific process model. This means that the controller can only be detuned (according to physical limitations on the final control elements) and never tuned very aggressively. For example, a valve opening speed can never be infinite; therefore, the value of R can never realistically be zero. It is known and discussed by many authors (e.g., Shinskey [24]) that the disturbance rejection of industrial MPC controllers lags behind that of PID controllers when they are specifically tuned for disturbance rejection. Recent MPC improvements in the area of state update [20] have closed that performance gap if the observer model is assumed to be known perfectly. However, in the presence of model mismatch, the control performance (IAE) of a PID controller is still better than that of an MPC controller with the best possible tuning (see Figure 2.10).

Lundstrom, Lee, Morari and Skogestad [21] discuss limitations of dynamic matrix control (DMC), such as suboptimal feedback performance for input disturbances and problems with instable processes. Their paper shows how MPC with an observer can be used to improve the feedback control performance. Some of the limitations discussed by Lundstrom, et al will become apparent in following chapters. When DMC is compared to MPC with observers, usually a large performance gap can be observed

(see Figure 2.11). Though observers improve MPC feedback performance, they still have assumptions that empirically tuned controllers, such as PID, do not have. Any model-based predictive controller and model-based observer will assume that the model is known perfectly. Even small model errors can cause large prediction and state update errors. Various correction or filter methods are utilized to reduce the impact of model error, but obviously optimality cannot be guaranteed if model error occurs. Even though many industrial processes are minimum phase, the majority of closed loops are not. Time delay, also known as deadtime, and higher-order lags create right hand poles that greatly complicate tuning. In most instances, closed-loop deadtime is created by transport delay of material in pipes and discrete sampling mechanisms that are unavoidable in computer control systems, while higher order lags are usually a result of filter time constants in measuring and control devices.

Other challenges often found in industrial plants are resolution and deadband problems created by the mechanical behavior of valves and packing. Mechanical limitations lead to stick-slip behavior, which causes non-linearities for small moves in manipulated variable. These limitations present many challenges to control engineers in industrial plants. Even if a certain process is expected to act like a first-order filter with certain gain and time constant depending on vessel geometry, one has to consider additional time constants from transmitters, control elements computer sampling, and jitter. Usually one should not rely solely on first principles models that may be available for some of the reactions. Since any digital control system has CPU and communication constraints, ample oversampling is not practical for all types of loops in a plant. A sampling rate of 1/5 of the dominant time constant is often considered reasonably sufficient. McMillan et al. discuss the impact of deadtime and sample period

on performance [11]. Since a controller can neither sense nor compensate for a load disturbance until after one loop deadtime, the minimum peak error is the exponential response after one deadtime. The process time constant τ_1 sets the exponential response. Thus, the peak error E_x for a loop tuned for max load rejection that takes into account the additional deadtime from the sample delay θ_s and the original deadtime θ_o is:

$$E_x = (1 - e^{-[(\theta_o + \theta_s)/\tau_1]}) \tag{1.25}$$

For $\theta << \tau_1$, where $\theta = \theta_o + \theta_S$, the minimum absolute integrated error can be approximated by the peak error E_x multiplied by the squared total deadtime θ:

$$IAE = \frac{\theta}{\tau_1} \theta * E_x \tag{1.26}$$

McMillan further suggests a scan period of at least 1/10 of deadtime or time constant for any controller that is tuned for feedback control performance, i.e. for optimal load disturbance rejection [14]. However, ample oversampling is usually not achievable for all loops in a plant. Specific examples include flow and pressure loops because they are inherently fast. Ideally, process model identification (and loop tuning) is performed by integrated automatic tools [15] because first principles modeling and universal third party solutions that identify a process model by connecting directly to the field instrument are not integrated, and thus do not consider or only approximate the effect of the computer control system itself on loop performance.

Many researchers and practitioners have been and are now actively involved in the development of new techniques that target solving problems that are encountered when trying to apply new or existing control strategies to actual industrial production

plants. Control conferences have been well-attended for decades, and there is an abundance of new control papers and methods every year. The constant development and improvement of modeling and control strategies makes the popularity of PID, a more than 100 year-old control strategy, even more remarkable. Only few control strategies that are alternatives to PID have actually found traction. Many MPC and fuzzy logic based PID replacement controllers have fallen out of fashion.

The following innovations were targeted at solving some of the issues described above. Addressing a great MPC weakness in 2002, Badgwell & Muske developed a method to augment disturbances to the model matrix, thereby better accounting for unmeasured dynamic disturbances in the feedback path [20]. This approach assumes that the perfect process model is known. In 2001, Shinskey improved the deadtime handling of PID control as he recognized this as a significant weakness in single loop control[25]. His method makes PID control applicable to deadtime-dominant processes, while simultaneously retaining very good feedback performance. However, it also sacrifices robustness, one of the great benefits of PID control. An example of a new PID replacement controller is Pannocchia, Laachi, and Rawlings' "Candidate to Replace PID Control - Constraint LQ"[41], developed in 2003, which exhibits very good setpoint tracking performance. This algorithm is tailored for minimum IE if input constraints reached; however, it also assumes the presence of a perfect model.

The research presented in this dissertation introduces two novel approaches for improving MPC control performance. Chapter 3 discusses a new automated calculation method for optimal tuning that is based on model mismatch, and Chapter 4 describes how the method described in Chapter 3 can be used for closed loop adaptive control. Chapter 5 introduces structural modifications to the MPC equation that improve the

disturbance rejection performance in the presence of model mismatch. Some of the PID properties that are most likely responsible for its unmatched popularity will be utilized to improve the MPC algorithm.

Chapter 2

State update methods and tuning

Model predictive controllers, like any model-based controller, are designed to solve feed-forward specific control challenges, such as difficult process dynamics and loop interaction. Therefore, it does not seem surprising that controllers designed only for feedback control, such as PID, can perform better on processes that require feedback correction. Feedback control is required whenever the controlled processes is affected by unmeasured disturbances, has model mismatch, or exhibits nonlinear behavior. Mismatch between model-based controller tuning and the actual process can be encountered due to identification error or process drift. All of the listed conditions exist in industrial processes. Thus, practical controllers have some method of accounting for these conditions (i.e., feedback).

In a model predictive controller, the state update (or state correction) algorithm is the mechanism that realizes control feedback as described below. The term "state feedback" is used to describe the process of updating the state variable based on process inputs and outputs. It is important to note the *difference between* state feedback and control feedback described above. An "optimal state update," as discussed in many previous research papers, does not necessarily translate into optimal feedback control performance. Control performance largely depends on the dynamics of the control error. Figure 2.1 depicts the schematic diagram of a predictive controller with embedded state observer:

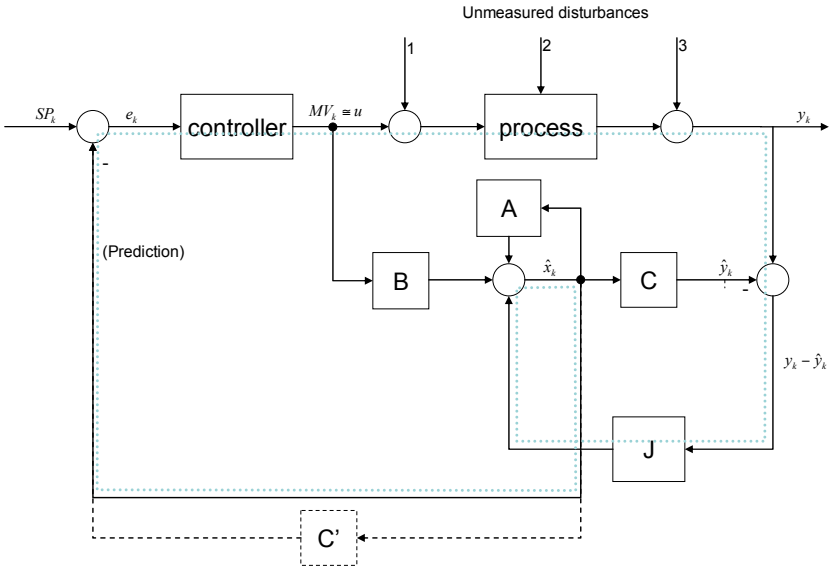

Figure 2.1: predictive controller with integrated observer; dotted line indicates feedback path; A, B and C are constant matrices that are used to represent the process model; J is the observer gain matrix

Based on the error e_k, the controller computes the optimal future outputs of manipulated variables MV_k over the control horizon. Ideally, the error vector is calculated directly as the difference between future prediction vector $\hat{x}$ and setpoint (SP) trajectory. However, the direct utilization of state variables is only possible in controllers that are realized in terms of step response process (SR) models. According to Qin and Badgwell[32], dynamic matrix control (DMC)[1] is the most commonly used controller of that kind. In the case of minimum realization model-based control, such as state space (SS), the future output prediction vector is not readily available and has to be re-computed at every scan from the model (dashed line and C' in Figure 2.1).

Oftentimes an observer (Kalman filter) is used to correct the state variables for conditions such as the disturbances as described above. Figure 2.1 shows the implementation of this observer with Kalman filter gain, which is pre-calculated based on what is known about the model before the controller is commissioned, i.e., it is a constant during normal operation as will be discussed in the following sections of this chapter. Assuming a constant gain matrix for J, it becomes apparent from Figure 2.1 that the feedback path between measurement and model-based controller introduces no dynamic control elements and acts as a proportional only controller (P-controller). Even though the calculation of the optimal J is mathematically refined, a P-only state feedback may not necessarily translate into optimal feedback control performance.

State of the art state estimation theory can provide offset-free control for positional MPC (e.g. Pannocchia and Kerrigan[44]). With incremental MPC controllers such as dynamic matrix control, offset-free control is inherent because the incremental controller equation:

$$dMV = \overline{\overline{K}} \otimes \bar{e} \tag{2.1}$$

acts as an integrator with

$$MV = \int_{t_0}^{t+p} \overline{\overline{K}} \otimes \bar{e}\, dt + MV_0 \tag{2.2}$$

where $\overline{\overline{K}}$ is the MPC controller gain matrix and $\bar{e}$ is the error vector calculated from future prediction and the setpoint (SP) trajectory. The integration of the error vector *e* occurs at every scan over the entire prediction horizon p. This shows that the closed-loop control path as indicated by dotted line in Figure 2.1 can thus be considered

to incorporate a PI controller. However, this dynamic matrix based PI-controller is still different from the traditional PI-controller described by

$$u(t) = K_P \left[e(t) + \int_{t_0}^{t} K_I e(e) dt \right] \tag{2.3}$$

because here an error scalar is integrated. In the settled case (no transients) the I-term of a PID may be a nonzero constant, while the I-term of an MPC is zero at the unconstrained steady state.

This chapter discusses the importance of state update in predictive controllers and the optimal tuning of state update. Optimal state update in a model predictive controller does not necessarily lead to optimal control performance, which will be shown in the following chapters.

2.1 Modeling Techniques

State space (SS) models have become popular in the recent years. One of the many advantages of SS models is that they allow minimum realization of models, which reduces dimensionality of the model matrix when the process deadtime is relatively small compared to the process time constants. This translates to a reduction of states in state variable $\hat{x}$. However, step response (SR) models are still preferred by many users because they are more intuitive and can construct the future prediction vector in a natural way. Commercial products usually display the future prediction to operators and/or use it to detect and alarm future limit violations. For model predictive controllers, this optimization normally means a much smaller calculation effort (CPU time) which easily makes up for the higher number of states (memory storage). Another practical advantage of step response models is that a multitude of identification

techniques exist, that can compute them directly. Step response models can be shown to be a special case of the more general state space model. Lee et al. [31] compute the state space model parameters as:

$$A = \begin{bmatrix} 0 & I_{ny} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & I_{ny} & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & I_{ny} & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & I_{ny} & I_{ny} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_p & C_d \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & A_d \end{bmatrix};$$

$$B = \begin{bmatrix} SR_1 \\ SR_2 \\ \vdots \\ SR_n \\ SR_{n+1} - SR_n \end{bmatrix};$$

$$C = [I_{ny} \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad 0]$$

where $SR_i = \begin{bmatrix} s_{1,1,i} & s_{1,2,i} & \cdots & s_{2,n,i} \\ s_{2,1,i} & s_{2,2,i} & \cdots & s_{2,n,i} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m,1,i} & s_{m,2,i} & \cdots & s_{m,n,i} \end{bmatrix}$ is the step response model matrix and A is the matrix that determines which step responses are integrating vs. stabile. Such state space models have explicit future prediction state vectors for each process output that form an n x p prediction matrix. This matrix can be modified by using very easy geometry. Terry Blevins et al. [11] presents one of the variations of this concept, as outlined below:

$$x_{k+1} = Ax_k + B\Delta u_k + Jw_k \tag{2.4}$$

$$y_k = Cx_{k+1} \tag{2.5}$$

where $x_k = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix}$ is the process output prediction at a time k, $\Delta u_k = u_k - u_{k-1}$ is the vector of change on the process inputs including measured disturbances $w_k = y_k - \hat{y}_k$ is the vector of residuals, J is the n x p dimension filter matrix. The MPC controller updates prediction and control calculations every scan. This procedure is known as receding horizon control. If a state space model of this form, i.e., a step response model, is used in the control configuration as shown in Figure 2.1, the state variable x can be used directly to compute the MV moves over the control horizon. The effect of observer gain J then becomes a simple biasing of the state vector by weight j_i where $J_k = \begin{bmatrix} j_1 & 0 & \cdots & 0 \\ 0 & j_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & j_n \end{bmatrix}$.

As mentioned before, besides unmeasured disturbances, model mismatch is one of the reasons why feedback becomes necessary. One cannot assume that model parameters are known exactly. The well-known analytical Kalman filter can optimally update the state variable of a process characterized by a stochastic state-space model:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{2.6}$$

$$y_k = Cx_k + n_k \tag{2.7}$$

for Gaussian distributed process noise w_k and measurement noise n_k . However, one of the assumptions of the above equation is that A, B and C are known

exactly. This formulation may not applicable to processes with model mismatch. Every real process suffers from this problem.

Experience with PID tuning showed that model-based tuning, such as lambda tuning and IMC, although more sophisticated, is often incorrect because the underlying plant model is unknown, difficult to identify, or drifting. Many empirically derived tuning rules have been published. Methods that are based on critical frequency and critical gain, such as Takahashi [30] and Neural Network modified Ziegler-Nichols[45] tuning, perform better in such cases. Similarly, control vendors often apply empirical state variable correction schemes on their model predictive controllers [16]. As described in the previous section, in a dynamic matrix controller it is straightforward to reconcile state variables (future process output predictions) and measured process outputs by simple geometric manipulation of the state. The state update is usually an empirical manipulation based on the residuals. For self-regulating processes, simple biasing may be used while integrated processes require additional rotation of the future process output prediction. This implies that the engineer specifies which processes are integrating vs. stabile.

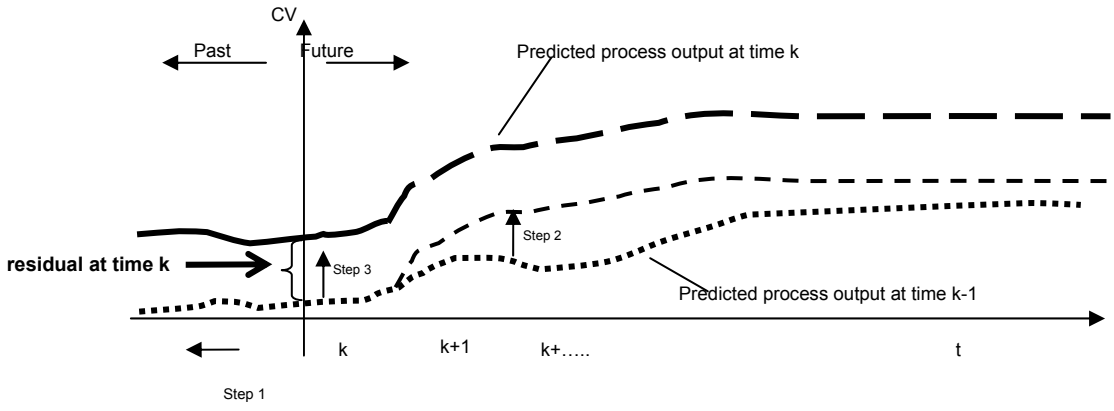

Figure 2.2: Prediction correction using bias term based on current measurement

Figure 2.2 shows the general feedback-based heuristic shifting approach in a dynamic matrix controller. The following steps are repeated at every scan:

Step 1: Time shift of entire prediction to the left

Step 2: Prediction update based on recorded process input changes (including measured disturbance variables)

Step 3: Correction of prediction based on measured residuals

One of the first efforts of this research project was to determine how effective current correction methods are. Most of the widely known and used technologies can actually be shown to have feedback performance similarly to a P-controller. Methods such as neural networks and expert systems have been developed to aid prediction of abnormal situations by estimating the state of complicated and/or nonlinear processes. Usually these technologies are used open-loop to alarm operators or trigger automatic actions if certain states are reached. Directly modifying the state vector of a model predictive controller in a nonlinear way can be used to close the control loop. If known nonlinear models exist for nonlinear and partially nonlinear processes, external prediction update can improve the control performance and constraint handling. One may draw conclusions on how to better incorporate state information from external sources into MPC from the information learned through research on more effectively updating the state of MPC. Thiele et al. [16] describe an approach that naturally improves the future prediction of the controller without modifying the control matrix. Differing from the state update described in the previous chapter, the external state modifier adjusts the state variable at the point that corresponds to the furthest point in the future as shown in Figure 2.3.

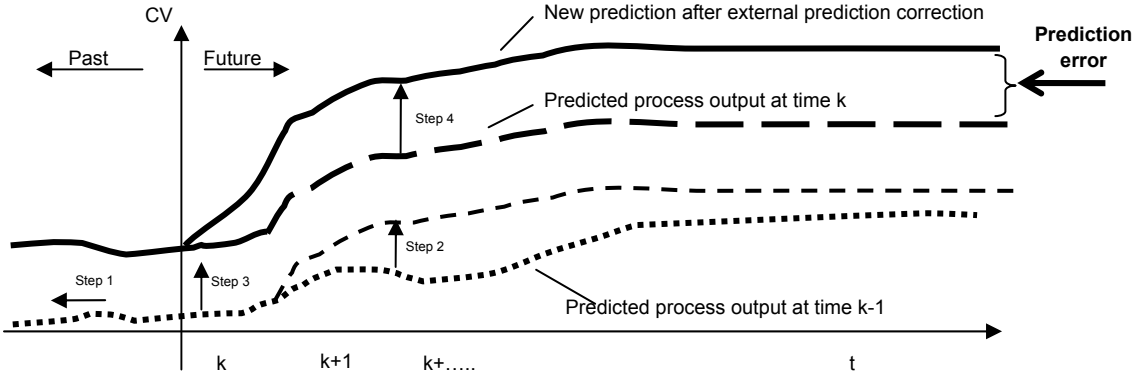

Figure 2.3: Prediction correction based on external future estimate from external state estimator

In this implementation, the external state modification is applied after the feedback-based modification (the three steps discussed earlier). An external algorithm, such as a neural network based "soft sensor", can supply the updated state information under in the following scenarios:

(1) If the absolute value of a process variable can be computed more exactly by a nonlinear algorithm in an external function block, this block may override the absolute prediction value directly.

(2) If the external block detects a certain condition or operation region it may inject a relative bias in the prediction, which is added to the dynamic linear prediction inside the MPC block.

It is likely that external predictor has a different time horizon than the MPC controller; thus, the described implementation allows injecting bias or absolute value at any point in the horizon. The MPC state update algorithm automatically extrapolates in this case. Ideally, the prediction correction is distributed over the horizon in a process

specific trajectory. Figure 2.4 depicts two different prediction modifications. While a linear prediction modification, as shown in Figure 2.4a, may be a suitable approximation for most process control challenges in the industry, a nonlinear distribution of prediction error over the prediction horizon, Figure 2.4b, can create a more drastic controller action where desired, e.g. control of polymer processes.

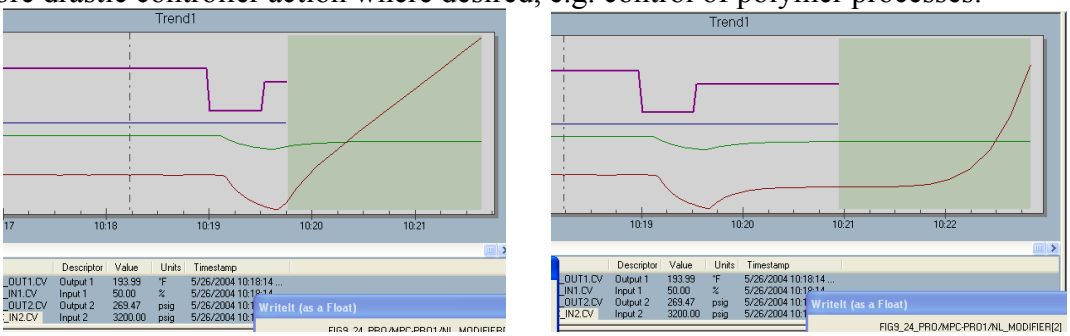

Figure 2.4: a- linear distribution; b- exponential distribution of feedforward adjustment

In recent years process simulators based on step response models [46] have become useful alternatives to well known process simulation packages such as Aspen[47] and HYSYS[48]. Step response model-based simulation is usually used when first principle or ODE process models are not available or too complicated to derive. The great advantage of universal mathematical step response models is that there are many tools available today that can automatically identify processes in a black box manner from historical data or bump tests. Some of the applications of step response based process simulators are operator training and off-line performance testing prior to control system commissioning. Real-time applications that are more involved in the actual process control include simulation of trajectories between sample points on gas chromatographs and temporary simulation of failed measurements in model predictive controllers.

2.2 Optimal state update with Kalman filter

As discussed in Chapter 1, the tuning of predictive controllers does not have to tradeoff setpoint tracking and disturbance rejection performance. Improving setpoint tracking does not sacrifice load rejection performance and vice versa. Setpoint tracking performance of predictive controllers is known to be very good, and MPC controllers usually outperform PID controllers at this task. However, while the PID controller can be tuned to sacrifice setpoint tracking performance to improve disturbance performance, the MPC cannot, and thus usually lags behind PID at disturbance rejection. In 2002, Badgwell and Muske [20] suggested a method that improved the disturbance rejection performance of MPC to perform as well as the setpoint change response if the disturbances are introduced at the end of the process. Such disturbances are known as output disturbances, and may be practically introduced in the measurement device itself. Figure 2.5 shows the three possible entry points of unmeasured disturbances:

1. At the beginning of the process: The transfer function of the disturbance model is equal to that of the process; also known as input disturbances.
2. At the end of the process: The transfer function of the disturbance model is 1; also known as output disturbances.
3. Anywhere in between 1 and 3: The transfer function of the disturbance model is different from that of the process but not the same as 1.This is the most likely scenario in industrial processes.

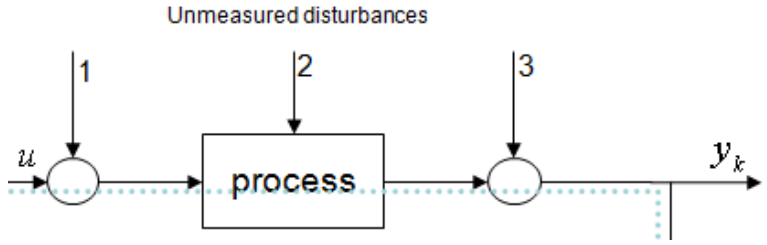

Figure 2.5: Entry points for unmeasured disturbances

If the unmeasured disturbances are introduced at the beginning of the process (input disturbances) they will have to travel through the entire process before they can be detected and corrective action can be taken. Unwanted deviation from the setpoint is unavoidable in this case. As discussed in the previous chapter, it will take one full deadtime before the controller can correct any unmeasured disturbances and the IAE can be calculated as:

$$IAE = \frac{\theta}{\tau_1}\theta * E_x \tag{1.26}$$

The presented research focuses on input disturbances, as these are the most difficult challenges to a feedback controller; the two other disturbance introduction points are a reduced severity subset of input disturbances. The general objective of state observers is to provide an estimate of the internal states of a system based on all measurable system inputs and outputs. Observer gains can be computed if a mathematical model of the system is known. The filter formulation developed in the 1960s by Kalman [26] has been the most popular method in process control for estimating internal process states based on noisy or incomplete measurements. For the discrete sampling system MPC formulation given in (1.1) to (1.3) the Kalman filter equation for estimating x_{k+1} is:

$$\hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k + J(y_k - \hat{y}_k) \tag{2.8}$$

$$\hat{y}_k = C\hat{x}_k \tag{2.9}$$

where J is the Kalman filter gain, x_k is the state vector with k state variables, y_k is the predicted process output and $\hat{y}_k$ is the actual value of the process output. If covariances for unmeasured disturbances and measurement noise are known, the general Kalman filter structure can be created by augmenting G_w (disturbance and noise model) to the plant model and then re-computing the MPC controller gain for the augmented model as shown in Figure 2.6.

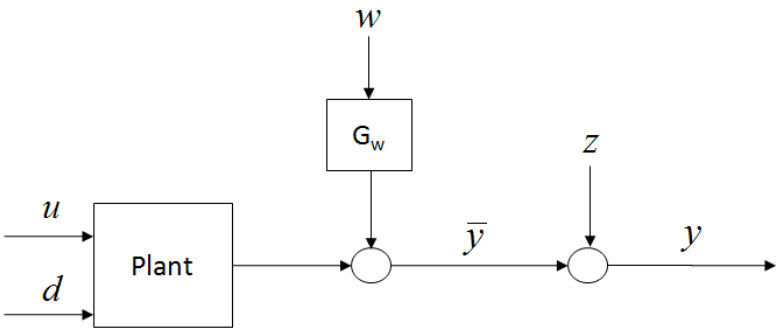

Figure 2.6: Augmenting G_w (disturbance and noise model) to the plant model

At that point the Kalman filter gain J can be calculated by numerically solving the Riccati equation which uses Q_{KF}, the positive semi-definite matrix representing the covariances of the disturbances in w and R_{KF}, the positive definite matrix representing the covariances of the measurement noise, z [26]. If the covariances are not known, Morari and Ricker suggest using a simplified version of the Kalman filter [27]. This formulation assumes that the disturbances are independent and thus each element of w affects one (and only one) element of y. Q_{KF} and R_{KF}, the input and measurement noise

covariances, are not required. Instead, this simplification uses a filter time constant τ_i and an estimate of the signal to noise ratio SN_i per disturbance to create the disturbance model as follows:

$$G_{W_i}(q) = \frac{1}{q - a_i} \tag{2.10}$$

where $a_i = e^{-\frac{T}{\tau_i}}$, $0 \leq \tau_i \leq \infty$ and T is the sampling period. As $\tau_i \rightarrow 0$, $G_{w_i}(q)$ approaches a unity gain, while as $\tau_i \rightarrow \infty$, $G_{w_i}(q)$becomes an integrator. Element i of Δw is a stationary white-noise signal with zero mean and standard deviation σ_{w_i} (where $w_i(k) = w_i(k) - w_i(k - 1)$). Element i of z is a stationary white-noise signal with zero mean and standard deviation σ_{z_i}.

2.3 The control performance aspect of state update

As discussed in 2.1, the objective of state update is to find the best possible estimate of the current state variable at every instance of time (i.e., at every scan period of a discrete controller). Utilizing the best possible state estimate in a well-tuned MPC controller does not necessarily mean that it will lead to the best possible control performance.

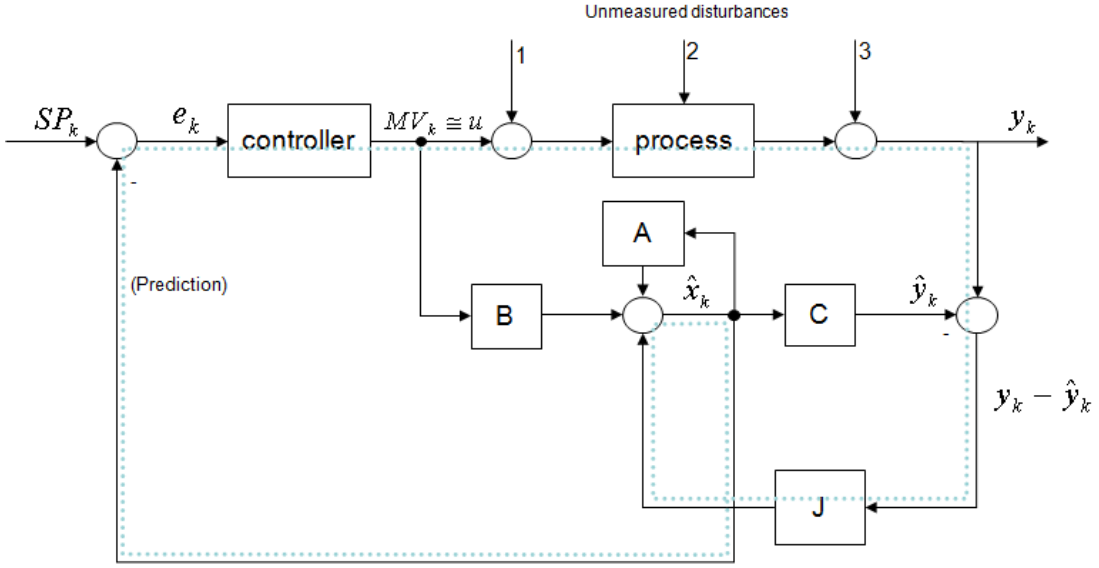

Figure 2.7: State update in closed-loop

Figure 2.7 depicts the closed control loop represented by controller, process and state estimator (2.8)/(2.9):

$$\hat{x}_{k+1} = A\hat{x}_k + B\hat{u}_k + J(y_k - \hat{y}_k) \tag{2.8}$$

$$\hat{y}_k = C\hat{x}_k \tag{2.9}$$

The closed loop feedback path is represented by a dotted line. It becomes obvious that the dynamic behavior (i.e., transfer function) of the closed loop depends on J. Since the observer gain J is derived from noise covariance or signal to noise ratio in the case of a Kalman filter formulation (as shown in section 2.1), there is no tuning parameter or generic variable that takes the observer transfer function into account. Therefore, the closed-loop control performance may be affected in an undesirable (sub-optimal) way. Other tuning methods for J that are used in the industry include bias correction, rotation factors and different types of Kalman filter.

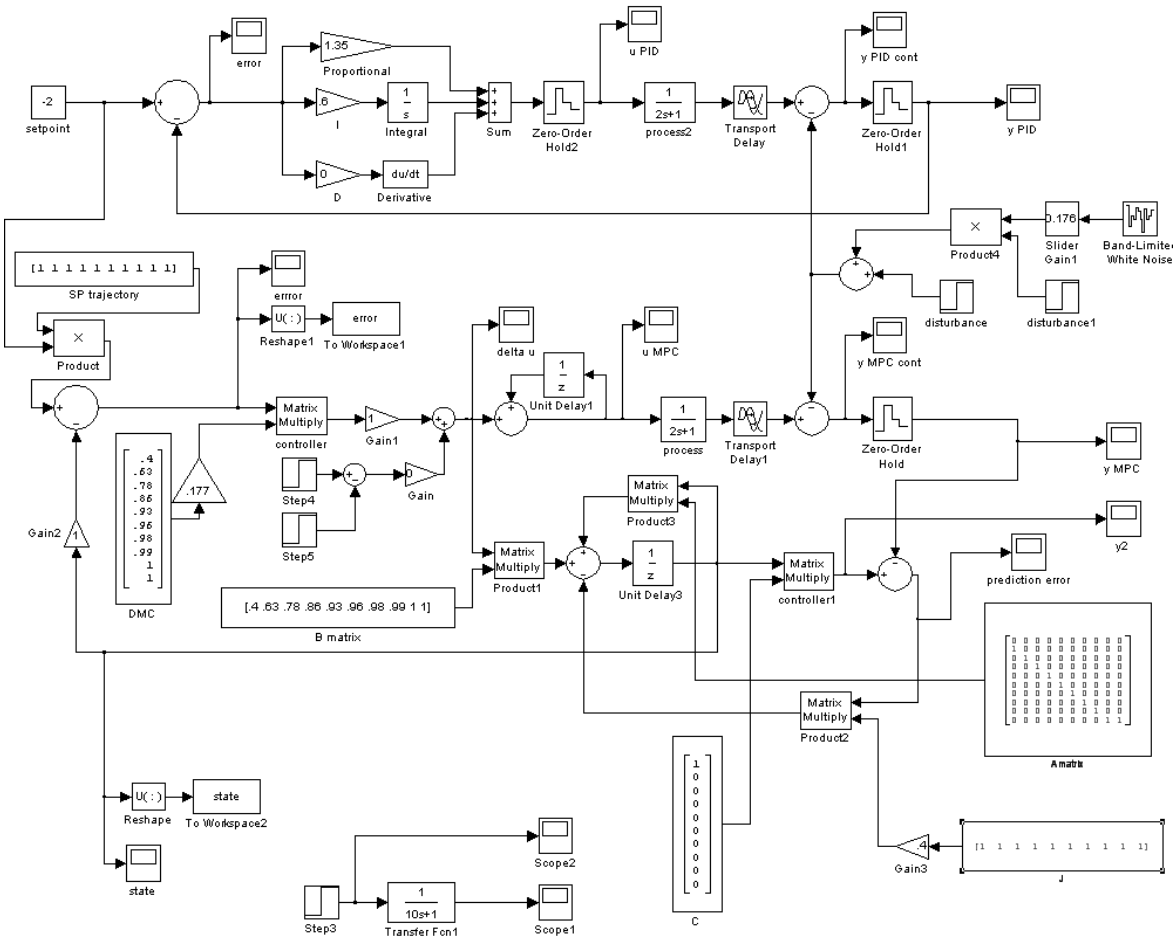

Figure 2.8: Block diagram of MPC observer and process simulation in Simulink®

A simulation that corresponds with the closed loop schematic in Figure 2.7 was built by the author to verify the impact of J on control performance. The block diagram including process simulation is depicted in Figure 2.8. While a model predictive tool box with integrated MPC controllers is already available for MATLAB, all MPC and observer features are coded in the MATLAB structured text programming language. Further, the Simulink® simulation, shown in Figure 2.8, was implemented to be able to easily display and modify specific components of controller and observer. This

particular schematic diagram shows a DMC implementation. Thus, the A matrix is diagonal and the observer gain J is unity gain, as suggested by Lundstrom et al. [21]. This simulation was used to create the responses described in Figure 2.9 and Figure 2.10. The same results can be obtained by using the MPC toolbox function cmpcsim.

Figure 2.9 shows the reaction of MPC control with a different observer gain J to a setpoint change (from 0 to -2 at t=0) and a change in unmeasured disturbance (from 0 to 2 at time t=20) when connected to a process with transfer function $G(s) = \frac{1}{2s+1}$. The response of a PI controller to an identical stimulus is added for reference.

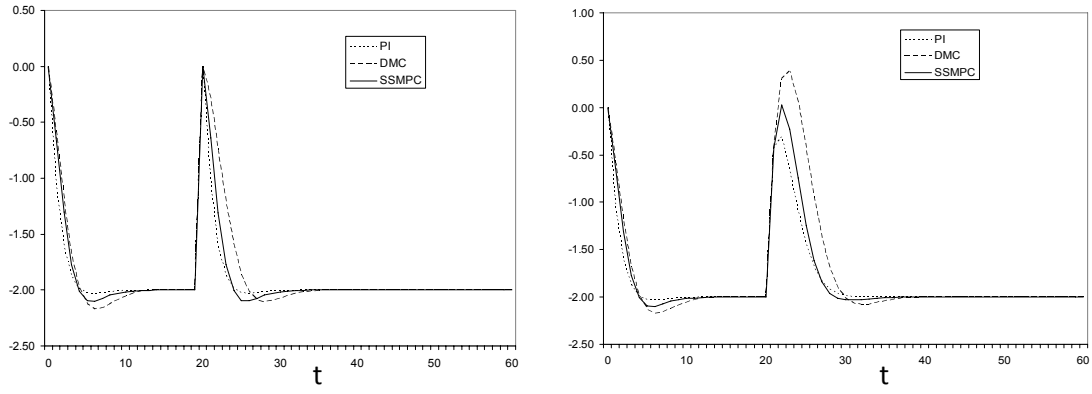

Figure 2.9: Setpoint change response and load disturbance rejection of controllers with different observer gains for input and output disturbance: general state space MPC (SSMPC) with J=[0 0 0 0 0 0 0 0 0 1]^T, MPC with prediction biasing (DMC) with J=[1 1 1 1 1 1 1 1 1 1]^T, PI with Kc=1.35, Ti=1.7

Figure 2.10 depicts the response for the identical controllers when applied to a process with transfer function $G(s) = \frac{1}{4s+1}$, i.e., model mismatch ratio of $\tau_1 / \tilde{\tau}_1 = 2$.

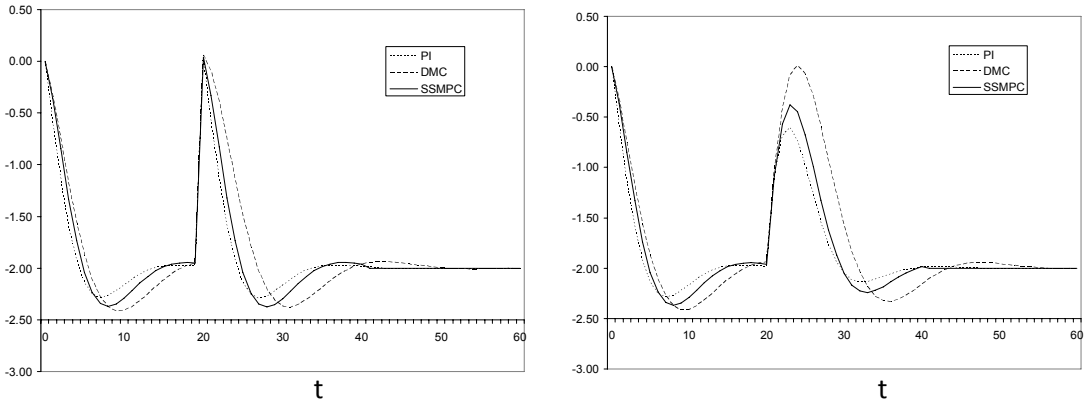

Figure 2.10: Setpoint change response and load disturbance rejection of controllers with different observer gains for input and output disturbance with process model mismatch $\tau_1 / \tilde{\tau}_1 = 2$: general state space MPC (SSMPC) with J=[0 0 0 0 0 0 0 0 0 1]^T, MPC with prediction biasing (DMC) with J=[1 1 1 1 1 1 1 1 1 1]^T, PI with Kc=1.35, Ti=1.7

When the unmeasured disturbances are applied on the process output PI and SSMPC, controllers perform very similar—setpoint tracking and load rejection performance are both very good. The same results can be observed for both of the processes described, i.e., with and without model mismatch. Dynamic matrix control (DMC) performed slightly worse during load changes. When the unmeasured disturbances are applied on the process input, PI control showed better load rejection than either of the two MPC controllers, independent of how the observer gain was derived. In further simulations no observer gain J could be found that improved load rejection performance significantly.

It should be noted that the PI controller used for reference here is not ideally tuned for load rejection, but rather for a compromise between setpoint and load rejection performance (as described in section 1.4), so that the setpoint change performance is comparable with that of MPC. The tuning parameters are K_c=1.35,

T_i=1.7. When the PI was tuned with Skogestad rules at K_c=25, T_i=8 for optimal load disturbance rejection (thereby sacrificing setpoint change performance), it performed better than MPC with the best possible observer gain J.

Since the closed loop responses for a large range of J are very similar, it seems that the value of J only has a very small impact on control performance. Surprisingly, this observation holds true for both scenarios: perfect model and model mismatch. Tuning of move and error penalties has a much larger impact on control performance with or without model mismatch, as will be shown in the next section.

2.4 Design parameters and their effect on control performance

This section analyzes how control performance is degraded by model mismatch and the effect that tuning of the controller parameters has in this case with and without model mismatch. A unit step disturbance Δw=1 is introduced at t=0 to three single loop MPC controllers at the input of the process as shown in Figure 2.11 and Figure 2.12. The controllers are only different in their state update mechanism. The three state update options as discussed in section 2.2 are:

1. Bias update as proposed in the original dynamic control (DMC) by Cutler [1]
2. General Kalman filter with tuning parameters Q_{KF} and R_{KF}
3. Simplified Kalman filter with tuning parameters τ_i and SN_i by Morari and Ricker [27]

A PID controller with tuning that emphasizes disturbance rejection (Skogestad) is also analyzed as reference. The Integrated Absolute Error (1.15), which is a very

popular criterion for control performance in industry, is used as a means to compare the four controllers. The controllers are regulating a second order process with time delay that has the following transfer function: $G(s) = \frac{1}{(30s+1)(20s+1)} e^{-s}$. The remaining tuning parameters are fixed at P=30; M=9; Q=1 (MPC) and K_C=52.5; T_i=28; T_d=5.7 (PID, with Skogestad tuning). Figure 2.11 shows the effect of the above mentioned step input disturbance on the controlled variable (process output) over time. The MPC controller with general Kalman filtering for state update clearly demonstrates the best control performance with IAE = 0.2 (more than 15 times better than DMC). The PID controller performance falls in between that of the two MPCs that use Kalman filtering. This suggests that MPC with general Kalman filter is the formulation that fits this particular process in the optimal sense.

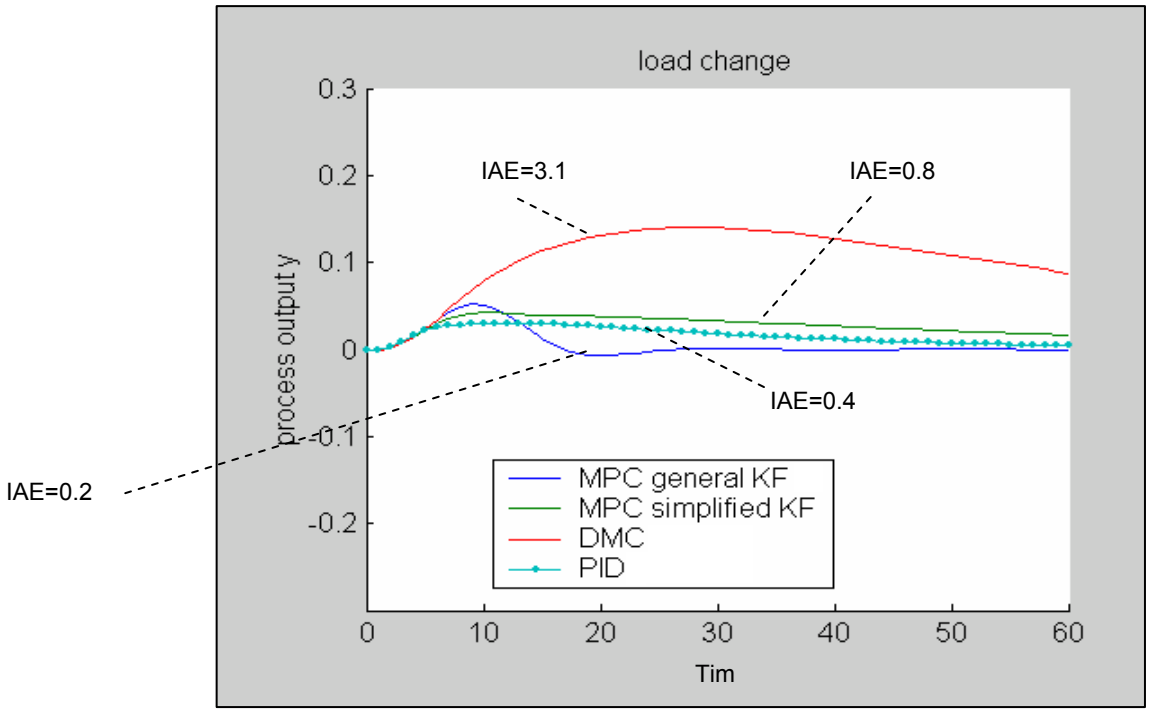

Figure 2.11: Disturbance rejection: controlled process output parameter as function of time

When various degrees of model mismatch are introduced in the same loop (Figure 2.12), it becomes apparent that this formulation is also the least robust. To normalize model mismatch is defined as the ratio of assumed (in control equation) process parameter to actual process parameter: $\tau_1 / \tilde{\tau}_1$

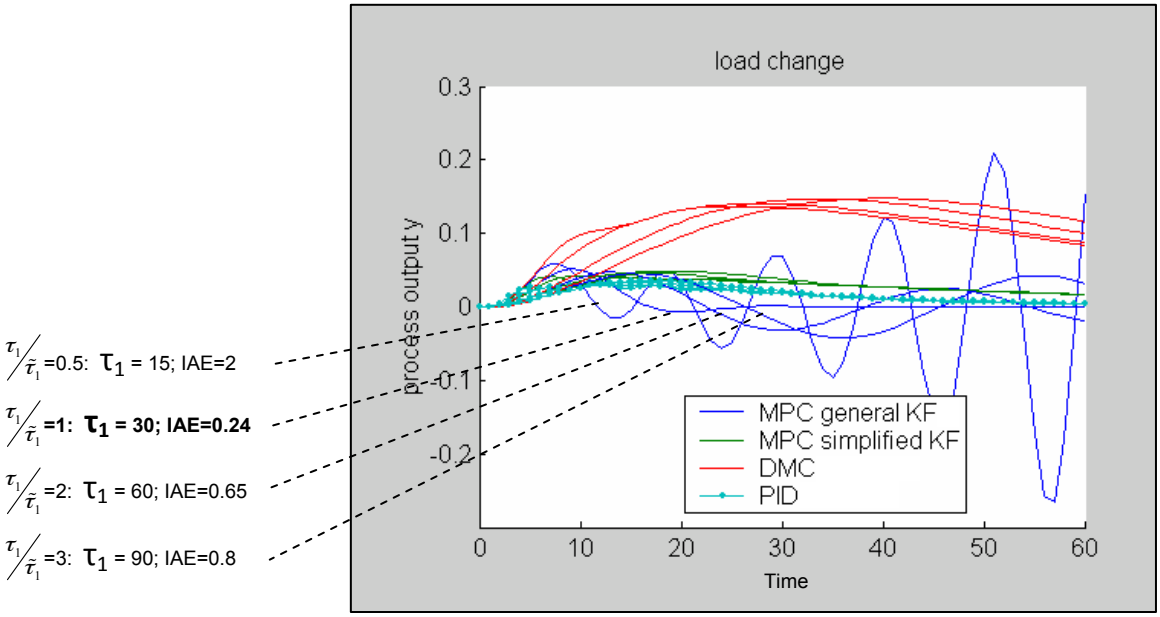

Figure 2.12: Effect of model mismatch in τ_1 (2^{nd} order process)

According to equations (1.1) to (2.9), the design parameters of a model predictive controller with Kalman filter are Q, R, S and J, where J depends either on Q_{KF} and R_{KF} for the general Kalman filter approach or on τ_i and SN_i for the simplified Kalman filter approach. Additional design parameters of a discrete MPC controller are prediction horizon P and control horizon M. An engineer may tune all of the above parameters to achieve desired control behavior that best fits a particular application.

The most frequently tuned parameters are the penalties (Q, R, S). They are normally tuned with process restrictions, such as different speeds of control or known model mismatch, in mind. In an industrial plant unit, different manipulated variables may have different requirements with respect to how fast they react to a setpoint change or a process disturbance. In some cases, variables are intentionally "clamped down" in order to remove oscillation from large control elements. McMillan investigates the impact of penalty tuning and gives an example of how to reduce overall variability by optimizing a coarse valve / fine valve problem with MPC [28]. Figure 2.13 shows how different values of R effect the control performance of three different single loop MPC controllers (the PI response to the same stimulus is plotted for reference – R does not apply).

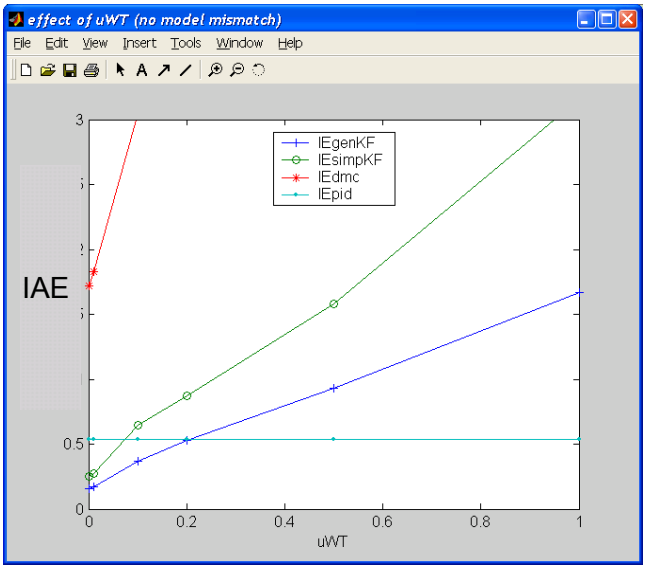

Figure 2.13: Effect of move penalty R on the integral absolute error of different controllers subject to an unmeasured disturbance of 1. Process: **K**=1; τ_1 =30s; τ_2=20s; Θ =1; MPC: P=30; M=9; Q=1; PID: K_C=52.5; T_i=28; T_d=5.7 (Skogestad tuning)

All trends indicate that the smallest possible R will result in the smallest IAE, and therefore the best control performance. This outcome makes sense because this

scenario has no model mismatch; increasing values of R will unnecessarily slow down the reaction of the controllers to unmeasured disturbances. This outcome also makes it obvious that the model predictive controller with general Kalman filter has the best performance, while DMC has the worst. These results are rather intuitive, because DMC does not have an observer that is matched to the process model dynamics G_W as described in 2.1.

If the model does not match the process, the effect of tuning the penalty changes drastically as shown in Figure 2.14.

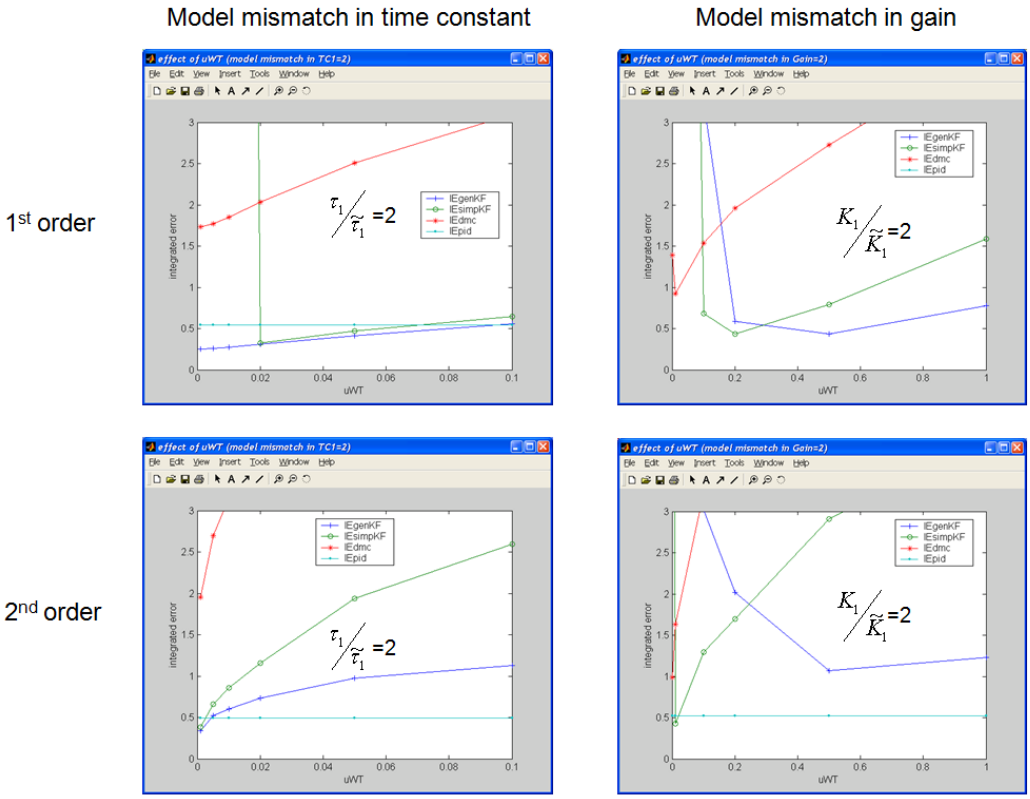

Figure 2.14: Effect of move penalty R in the presence of model mismatch on processes of different order. Process: **K**=1, τ_1 =30, τ_2=20s, Θ =1; PID: K_C=52.5, T_i=28, T_d=5.7 (Skogestad); MPC: P=30, M=9, Q=1

For model mismatch in gain, IAE becomes a convex function of R when a Kalman filter is used. If the model mismatch is in time constant only, a different relation appears. Dramatic differences are also visible between first and second order processes. In most commercial MPC, controllers control horizon M and fix prediction horizon P to a certain value that is designed to be the best tradeoff between control performance and CPU usage for a specific implementation [32]. Figure 2.15 shows how MPC performance varies for different values of control horizon.

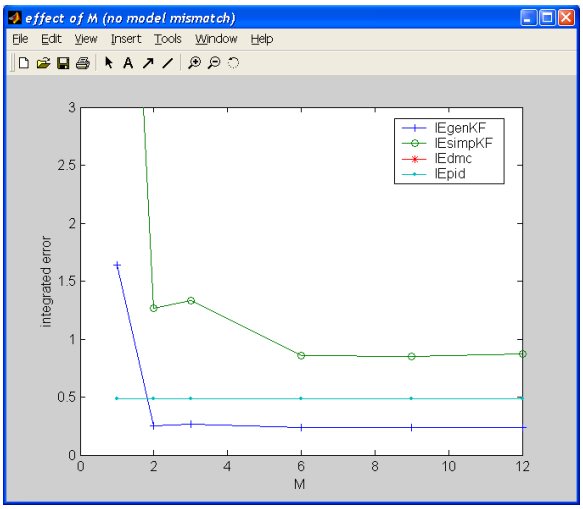

Figure 2.15: Effect of Control Horizon to control performance with perfect model on the integral absolute error of different controllers exposed to an unmeasured disturbance of 1. Process: **K**=1; τ_1 =30s; τ_2=20s; Θ =1; MPC: P=30; M=9; Q=1; PID: K_C=52.5; T_i=28; T_d=5.7 (Skogestad tuning), DMC is off scale

If a perfect model can be assumed, the control performance of all examined controllers increases when the size of the control horizon is increased. However, for this single loop scenario the control performance settles fairly early and cannot be improved any further if M is increased beyond 6. Multivariable processes with difficult process interaction and higher order models require longer control horizons but settle out in a very similar manner. When analyzing the prediction horizon P, one can observe

the same behavior (the resulting curve settles out at about P=10 in this example). While the minimum acceptable control horizon depends strongly on the number and amount of interaction of process variables, in a multivariable controller the minimum acceptable prediction horizon depends on the deadtime to time constant ratio. When MPC controllers are used in deadtime-dominant process applications, move blocking may be used to artificially extend horizons in time [29]. Figure 2.16 shows the effect of control horizon to control performance in the presence of model mismatch for first and second order processes.

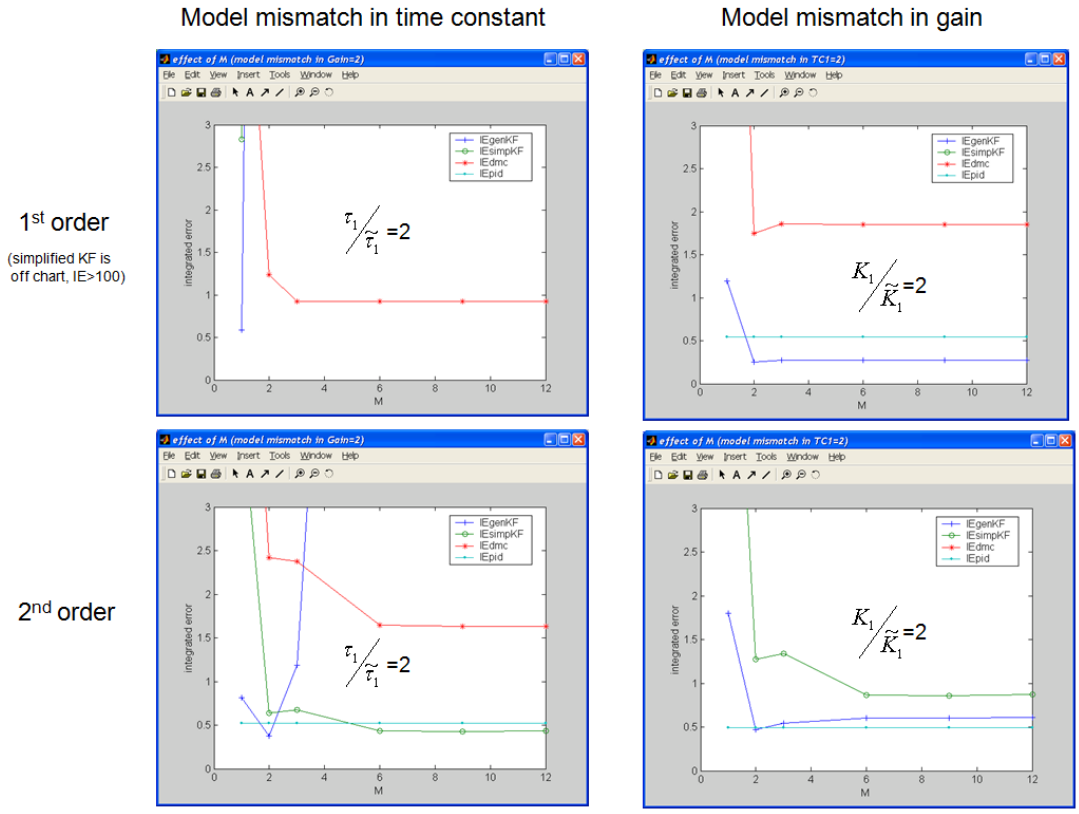

Figure 2.16: Effect of Control Horizon to control performance in the presence of model mismatch on processes of different order. Process: **K**=1, τ_1 =30, τ_2=20s, Θ =1; PID: K_C=52.5, T_i=28, T_d=5.7 (Skogestad); MPC: P=30, M=9, Q=1

As in the move penalty analysis, it becomes apparent that MPC with a general Kalman filter has very little tolerance for model mismatch. With a model mismatch ratio of 2, the control horizon has to be very small (1 or 2) to avoid oscillatory behavior, as shown in Figure 2.15. This is much lower than one would have usually designed just by looking at the behavior in Figure 2.15. The conclusion that can be drawn from this is that the amount of model mismatch must be known before a controller is tuned. With smaller model mismatch ratios (e.g., 1.5), larger values of M=3,4 were acceptable before instability occurred (not shown in figure). An interesting side observation is that DMC, the controller without an observer, performs better with model mismatch than without it. In addition, it outperforms the other controllers when the model mismatch is limited to the time constant. This fact suggests that not only the tuning, but also the ideal type of Kalman filter can only be selected with knowledge of amount and type of model mismatch (i.e., which model parameter). In Summary,

1. Optimal error and move penalties depend on which process parameter has model mismatch (gain, time constants, delay).
2. Optimal horizons become longer with higher model order and shorter with higher model mismatch.
3. Optimal tuning for first and second order processes is very different (including optimal Kalman filter method).
4. Many relations shown in the plots above are convex functions, suggesting that optimization algorithm could be used to find an optimum.

The next chapter describes how optimal tuning can be found using a solver for situations with and without model mismatch.

Chapter 3

Optimal control in the presence of model mismatch

As discussed in section 1.4, tuning parameters for model predictive controllers are commonly used to adjust the controller behavior in a way that is desirable for a particular plant application. For example, a certain desired speed of response may be set by tuning the move penalties R to a certain value. However, the expected behavior that is designed by the commissioning engineer will only occur if the model mismatch is insignificant, which is rarely the case in industrial plants. To account for the apparent model mismatch, practitioners often resort to iterative tuning until the desired behavior can be observed. This process is costly because it is very time-consuming, and it is difficult to cover all control and constraint scenarios on a running plant. Even if this method results in the desired plant behavior for the given model mismatch, one can expect the behavior to change if the magnitude of model mismatch changes. Furthermore, even if the amount of model mismatch and its variation is known, today there is no method to derive tuning from this information. This chapter shows how knowledge of model mismatch can be used to determine tuning for optimal control performance in the presence of constant or changing model mismatch. Even though some practitioners have tried to address this problem with empirical methods [8], it is rarely addressed in control literature because textbook control theory is usually derived for perfect models only.

3.1 Calculation of optimal tuning

In section 2.4, the effect of the most common MPC tuning parameters were discussed and documented. As was shown above, the way in which tuning parameters

influence the MPC control behavior and closed-loop control performance depends on the amount of model mismatch. In some cases, relationships are very significant and/or nonlinear. The following MPC tuning and design parameters will be used for further analysis:

- MPC controller tuning:
  - P (prediction horizon), integer
  - M (control horizon), integer
  - Q_{MPC} (move penalty), float vector
  - R_{MPC} (error penalty), float vector
- Type of Kalman Filter (TKF): General or Simplified, boolean
- Kalman filter tuning
  - General
    - Q_{KF} (covariance of the disturbances in w), float matrix
    - R_{KF} (covariance of the measurement noise, z) , float matrix
  - Simplified
    - T (filter time constants τ_i, float vector
    - SNR (signal-to-noise ratio for each disturbance), float vector

Different implementations of MPC may use additional tuning parameters, such as maximum move rate or reference trajectory [32]. However, such parameters are usually intended for specific needs of operators, and the resulting effects overlap with the above mentioned parameters. Even though other means of influencing the dynamic behavior of MPC exist, the majority of desired behavior can be addressed with the parameters described in this work.

Since model mismatch and the tuning parameters are highly correlated with respect to the closed-loop control performance, an optimization problem with constraints was formulated. Figure 3.1 shows a schematic overview of the problem to be solved: based on knowledge of model mismatch and change in model mismatch, ideal tuning parameters shall be derived.

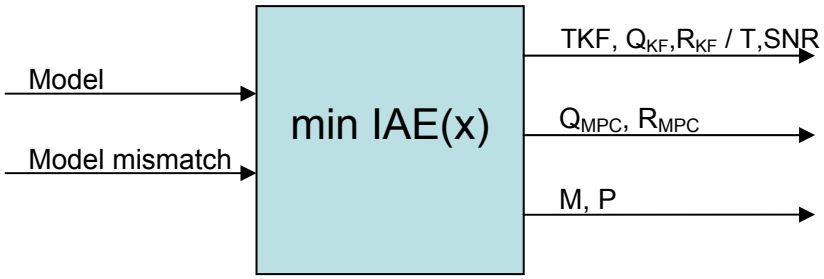

Figure 3.1: Schematic overview of optimization method: MPC tuning and design parameters are calculated based on model parameters and model mismatch

It is to be expected that the optimal set of tuning parameters for a given model mismatch is suboptimal for the scenario in which no model mismatch is present. Constraints have been added to the optimization to address physical and logical boundaries in an arbitrary way. As will be seen later, the exact value of the constraints influences only the range of calculation, not the overall result. The optimization problem can be represented as:

$$\min_i IAE(\Gamma, \Xi, {}^{\Xi}/_{\hat{\Xi}}) \quad \text{s.t. } g(\Gamma) \geq 0 \tag{3.1}$$

where $\Gamma = [P,M,Q_{MPC},R_{MPC},TKF,Q_{KF},R_{KF},T,SNR]^T$, $\Xi = [G, \tau_1, \tau_2]^T$, $g(\Gamma)$ are inequality constraints describing computational limits of the control algorithm and IAE is the control performance as introduced in Chapter 1. Figure 3.2 shows the best possible IAE for a given amount of model mismatch in process model gain, obtained by solving this optimization problem for different values of model mismatch. The detailed

optimization results are shown in Table 3.1. Active constraints are shown in bold. One can observe that the optimizer makes use of all tuning parameters to achieve the optimal control performance. It computes different tuning parameters for different values of model mismatch that result in fairly similar control performance, as long as no constraint is reached. Whenever a constraint is reached, the control performance suffers because the optimizer runs out of degrees of freedom (i.e., tuning parameters) to compensate for model mismatch. It also becomes apparent that MPC with general Kalman filter outperforms MPC with simplified Kalman filter if $K>\tilde{K}$, and MPC with simplified Kalman should be chosen if $K<\tilde{K}$. As seen in 2.1, the general Kalman filter formulation is more rigorous than that of the simplified Kalman filter, which uses EWMA filtering to update the state variable; thus, it cannot be tuned to deal with smaller-than-expected gain very well, but excels with larger-than-expected gain. Since MPC with simplified Kalman filter is based on filtering (section 2.1), it is more robust than MPC with general Kalman filter if the process response has a smaller than expected magnitude ($K<\tilde{K}$). However, for $K>\tilde{K}$ it produces a slightly higher integral absolute error than MPC with a general Kalman filter.

Note that the optimization method suggests relatively discontinues targets for Γ, the independent variables, for some values of $\Xi/\hat{\Xi}$, the model mismatch. This is most likely caused by presence of multiple local optima to which the particular nonlinear solver algorithm of this example implementation converges. No further investigation of different solvers or different solver settings was done because the same optima were found reproducibly in multiple runs and the resulting IAE values were already satisfactory and sufficiently continuous.

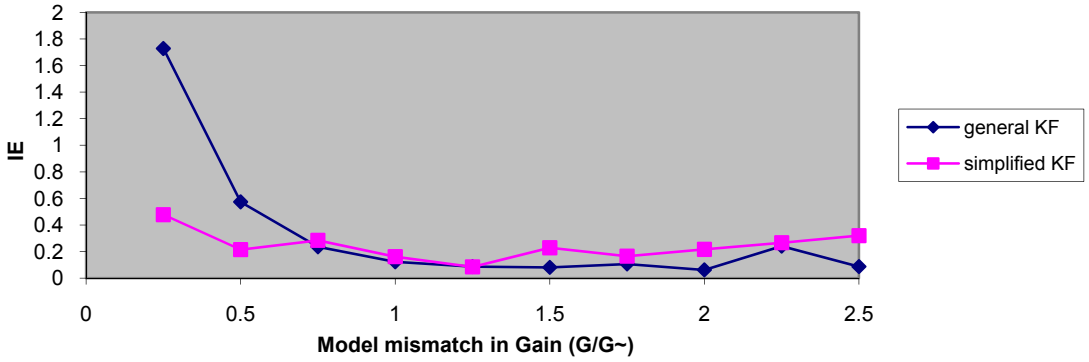

Figure 3.2 Optimal tuning of model mismatch in process gain

Model Mismatch	General Kalman Filter						Simplified Kalman filter					
	IAE	R_{MPC}	P	M	Q_{KF}	R_{KF}	IAE	R_{MPC}	P	M	T	SN
0.25	1.729	**0.0001**	**3**	**1**	**1**	**0.001**	0.478	**0.0001**	**3**	**1**	73.5067	30.0714
0.5	0.574	**0.0001**	**3**	**1**	**1**	**0.001**	0.215	**0.0001**	**3**	**1**	**100**	30.0916
0.75	0.236	**0.0001**	**3**	**1**	**1**	**0.001**	0.285	**0.0001**	**3**	**1**	20.3385	30.0104
1	0.123	**0.0001**	**3**	**1**	**1**	**0.001**	0.162	**0.0001**	**4**	**1**	**100**	30.0824
1.25	0.087	**0.0001**	**3**	**1**	**1**	**0.001**	0.085	**0.0001**	**3**	**1**	**100**	30.0788
1.5	0.081	**0.0001**	**3**	**1**	**1**	**0.001**	0.229	**0.0001**	**3**	**1**	12.6669	30.0055
1.75	0.107	**0.0001**	**3**	**1**	0.9946	0.0033	0.166	0.0095	15	6	**100**	30.0943
2	0.062	**0.0001**	12	6	**1**	0.0023	0.217	0.024	50	10	76.3204	30.1224
2.25	0.241	**0.0001**	3	**1**	**1**	0.0898	0.267	0.0424	23	19	**100**	30.048
2.5	0.087	**0.0001**	26	6	0.9974	0.0099	0.32	0.0769	15	16	**100**	30.2111
2.75	0.099	**0.0001**	27	6	0.9501	0.0182	18.182	**0.01**	27	11	10	30
3	0.111	**0.0001**	27	6	0.966	0.0312	16.667	**0.01**	27	11	10	30
3.25	0.123	**0.0001**	28	6	0.9355	0.047	15.385	**0.01**	27	11	10	30
3.5	0.135	**0.0001**	27	6	0.9981	0.0728	14.286	**0.01**	27	11	10	30

Table 3.1 Optimal tuning of model mismatch in process gain, active constraints are shown in bold

If the model mismatch is attributed to the first order time constant, the difference in integral absolute error between the two Kalman filter methods becomes more pronounced. As shown in Figure 3.3, if the process is more responsive than expected ($\tau_1<\tilde{\tau}_1$), the IAE rises with a very steep slope because oscillation occurs. Oscillation caused by $\tau_1<\tilde{\tau}_1$ in the time domain is depicted in Figure 2.12. Both methods

of Kalman filtering are affected similarly by oscillation, and an automatic method for optimal control in the presence of constant or varying model mismatch has to avoid it by all means. Details of such a formulation will be described later in section 3.3. However, if the process reacts slower than expected ($\tau_1 > \tilde{\tau}_1$), MPC with simplified Kalman filtering performs significantly better, which leads to the conclusion that general Kalman filtering, although stabile, should not be used in this scenario. Since the simplified Kalman filter formulation uses a filter time constant as one of the tuning parameters, an optimization method can easily use this tuning parameter to compensate for time constant mismatch in the plant. This compensation can easily be observed in the values of Figure 3.3. While the general Kalman filter tuning parameters are fixed at the constraints and only the MPC controller tuning parameters are still allowed to move, the T parameter of the simplified Kalman filter moves over a wide range and compensates for the model mismatch—thereby keeping the IAE at a very low level. For clarification, the Kalman filter type is specifically mentioned as a Boolean output of the optimization method described in Figure 3.1.

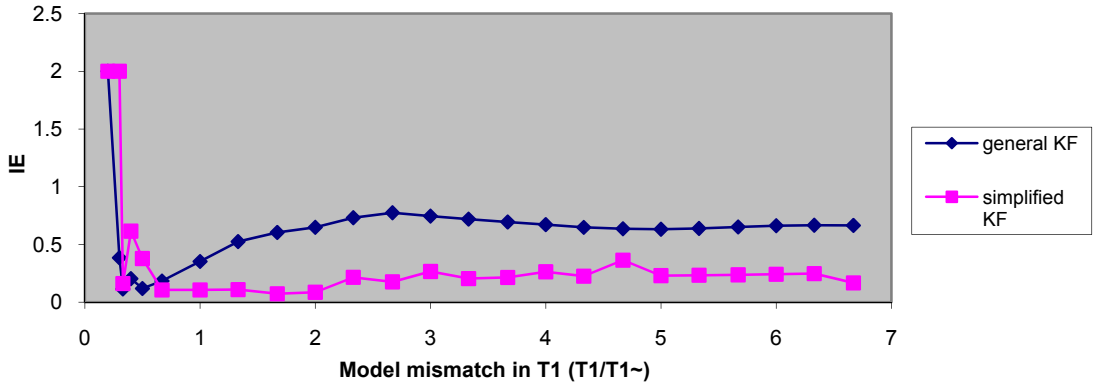

Figure 3.3 Optimal tuning of model mismatch in first order time constant

Model Mismatch	General Kalman Filter						Simplified Kalman Filter					
	IAE	R_{MPC}	*P*	*M*	Q_{KF}	R_{KF}	*IAE*	R_{MPC}	*P*	*M*	*T*	*SN*
0.2	2	0.01	27	11	**1**	0.2	2	0.01	27	11	10	30
0.3	0.385	**0.0001**	29	6	0.8569	0.0382	2	0.01	27	11	10	30
0.33	0.115	**0.0001**	**3**	**1**	**1**	**0.001**	0.162	**0.0001**	4	**1**	**100**	27.7497
0.4	0.204	**0.0001**	26	6	0.9962	0.0087	0.616	0.0466	13	12	**100**	30.0587
0.5	0.119	**0.0001**	12	6	0.9998	0.0021	0.379	0.021	**50**	8	71.2241	30.3909
0.67	0.184	**0.0001**	**3**	**1**	**1**	**0.001**	0.107	**0.0001**	**3**	**1**	**100**	30.0912
1	0.353	**0.0001**	**3**	**1**	**1**	**0.001**	0.107	**0.0001**	**3**	**1**	**100**	30.0908
1.33	0.525	**0.0001**	**3**	**1**	**1**	**0.001**	0.11	**0.0001**	**3**	**1**	96.8561	30.0866
1.67	0.604	**0.0001**	**3**	**1**	0.995	**0.001**	0.074	**0.0002**	**3**	6	72.0456	30.0819
2	0.649	**0.0001**	**3**	**1**	**1**	**0.001**	0.087	**0.0001**	25	6	63.5498	30.0362
2.33	0.732	**0.0001**	**3**	**1**	**1**	**0.001**	0.216	0.0001	**3**	**1**	20.2442	30.0104
2.67	0.775	**0.0001**	**3**	**1**	**1**	**0.001**	0.177	0.0001	**3**	**1**	50.0787	30.0427
3	0.746	**0.0001**	**3**	**1**	**1**	0.0026	0.267	0.0001	**3**	**1**	15.5521	30.0056
3.33	0.72	**0.0001**	**3**	**1**	0.9959	0.0018	0.205	0.0001	**3**	**1**	32.8161	30.0227
3.67	0.695	**0.0001**	**3**	**1**	**1**	0.0016	0.215	**0.0001**	**3**	**1**	27.6865	30.0514
4	0.672	**0.0001**	**3**	**1**	**1**	**0.001**	0.264	**0.0001**	**3**	**1**	15.3896	30.0051
4.33	0.648	**0.0001**	**3**	**1**	**1**	**0.001**	0.226	**0.0001**	**3**	**1**	24.2112	30.0134
4.67	0.636	**0.0001**	**3**	**1**	**1**	**0.001**	0.365	**0.0001**	**3**	**1**	10.0974	30.0001
5	0.632	**0.0001**	**3**	**1**	**1**	**0.001**	0.231	0.0001	**3**	**1**	22.9775	30.0294
5.33	0.639	**0.0001**	**3**	**1**	**1**	**0.001**	0.234	0.0001	**3**	**1**	22.5373	30.0165
5.67	0.652	**0.0001**	**3**	**1**	**1**	**0.001**	0.238	0.0001	**3**	**1**	24.6466	30.0142
6	0.663	**0.0001**	**3**	**1**	**1**	**0.001**	0.243	0.0001	**3**	**1**	22.5032	30.0125
6.33	0.667	**0.0001**	**3**	**1**	**1**	**0.001**	0.248	0.0001	**3**	**1**	21.2021	30.0126
6.67	0.665	**0.0001**	**3**	**1**	**1**	**0.001**	0.168	0.0001	12	20	26.4552	30.4974

Table 3.2 Optimal tuning of model mismatch in first order time constant

Another interesting observation that can be made when this optimization is solved is that the minimal possible IAE lies to the left of $\tau_1=\tilde{\tau}_1$ for the general Kalman filter. If the first order time constant of the actual process changes to about half the value that the MPC controller and general Kalman filter were designed for ($\tau_1 \sim 0.5\tilde{\tau}$), the IAE decreases, i.e., the recommended tuning for J (as per Riccati equation) does not yield the best possible control performance. This occurs because optimization problem (1.3) is designed to minimize static error while minimizing moves, while optimization

problem (3.1) is designed to minimize the IAE, thereby maximizing control performance.

The effect of model mismatch in the second order time constant shown in Figure 3.4 is very similar to that of model mismatch in the first order time constant shown above. The magnitude of the differences between general and simplified Kalman filters is smaller, however, the trend is the same.

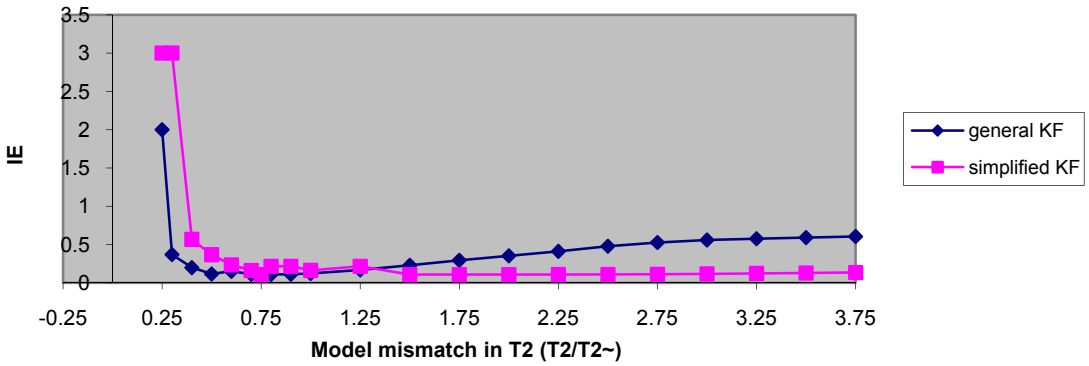

Figure 3.4 Optimal tuning of model mismatch in second order time constant

	General Kalman Filter						*Simplified Kalman Filter*					
Model mismatch	*IAE*	R_{MPC}	*P*	*M*	Q_{KF}	R_{KF}	*IAE*	R_{MPC}	*P*	*M*	*T*	*SN*
0.25	2	1.5182	44	6	**1**	**0.001**	3	0.01	27	11	10	30
0.3	0.37	**0.0001**	28	6	0.9454	0.0382	3	0.01	27	11	10	30
0.4	0.199	**0.0001**	30	6	0.996	0.0082	0.567	0.0404	12	12	85.7026	30.1881
0.5	0.116	**0.0001**	13	6	0.9971	0.0018	0.367	0.0176	29	8	**100**	30.434
0.6	0.151	**0.0001**	**3**	**1**	0.9918	0.0018	0.234	0.007	32	6	89.8552	30.1182
0.7	0.112	**0.0001**	**3**	**1**	**1**	**0.001**	0.161	**0.0001**	4	1	**100**	30.0948
0.75	0.11	**0.0001**	**3**	**1**	**1**	**0.001**	0.107	**0.0001**	**3**	**1**	**100**	30.0913
0.8	0.11	**0.0001**	**3**	**1**	**1**	**0.001**	0.215	**0.0001**	**3**	**1**	20.2633	30.0103
0.9	0.112	**0.0001**	**3**	**1**	**1**	**0.001**	0.215	**0.0001**	**3**	**1**	20.2576	30.0103
1	0.123	**0.0001**	**3**	**1**	**1**	**0.001**	0.162	**0.0001**	4	**1**	**100**	30.0824
1.25	0.169	**0.0001**	**3**	**1**	**1**	**0.001**	0.215	**0.0001**	**3**	**1**	20.2556	30.0103
1.5	0.229	**0.0001**	**3**	**1**	**1**	**0.001**	0.107	**0.0001**	**3**	**1**	**100**	30.0911
1.75	0.294	**0.0001**	**3**	**1**	**1**	**0.001**	0.107	**0.0001**	**3**	**1**	**100**	30.0912
2	0.353	**0.0001**	**3**	**1**	**1**	**0.001**	0.107	**0.0001**	**3**	**1**	**100**	30.0908
2.25	0.412	**0.0001**	**3**	**1**	**1**	**0.001**	0.108	**0.0001**	**3**	**1**	**100**	30.0908
2.5	0.478	**0.0001**	**3**	**1**	**1**	**0.001**	0.109	**0.0001**	**3**	**1**	**100**	30.0908

2.75	0.527	**0.0001**	**3**	**1**	**1**	**0.001**	0.111	**0.0001**	**3**	**1**	81.9354	30.0729
3	0.559	**0.0001**	**3**	**1**	**1**	**0.001**	0.116	**0.0001**	**3**	**1**	71.5816	30.0626
3.25	0.577	**0.0001**	**3**	**1**	**1**	0.0011	0.122	**0.0001**	**3**	**1**	74.8502	30.0667
3.5	0.591	**0.0001**	**3**	**1**	**1**	**0.001**	0.128	**0.0001**	**3**	**1**	62.3478	30.0663
3.75	0.605	**0.0001**	**3**	**1**	**1**	**0.001**	0.134	**0.0001**	**3**	**1**	58.3664	30.0445

Table 3.3 Optimal tuning of model mismatch in second order time constant

This section discussed how optimal tuning parameters for MPC control and state update can be obtained from knowledge about model and model mismatch by using optimization. The actual calculations and MATLAB implementation that was used for the examples above are shown in Appendix A. The impact of model mismatch and optimal tuning to compensate for it were analyzed by each model parameter separately. In a real plant scenario, all parameters that correspond to a prescribed model (and others that are not modeled for various reasons) will vary simultaneously. Which model parameter is most affected depends mainly on process type and cause of model mismatch (e.g., tube fouling, varying heat coefficient of fuel). Depending on the model identification method, a model mismatch in lead time constant may be interpreted as model mismatch in model deadtime or time constant. Often, one or two model parameters contribute significantly to model mismatch simultaneously. The next section analyzes model mismatch in multidimensional subspace.

3.2 Optimal tuning map

This section analyzes the effects of model mismatch in more than one model parameter and shows how optimal tuning can be calculated. Figure 3.5 depicts a surface plot of best possible IAE as calculated by optimization (3.1) shown in the previous section. Gain and first order time constants are allowed to have model mismatch. The optimal tuning for model mismatch in first order time constant and second order time constant are very similar, with the first order time constant being more significant (see

also Figure 3.3 and Figure 3.4); therefore three-dimensional visualization is preferred over four-dimensional, and the effects of model mismatch in second order time constant have been neglected in this plot.

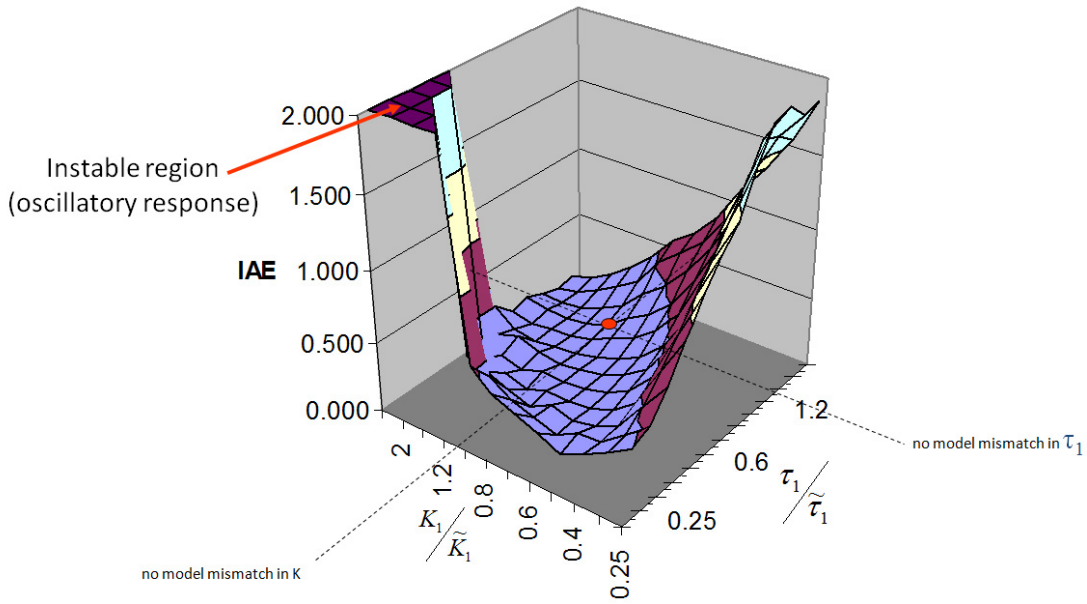

Figure 3.5: Optimal tuning map for model mismatch in K and τ_1 of MPC with general Kalman filter state update

For easier visibility the values of $K/\tilde{K}=1$ and $\tau_1/\tilde{\tau}_1=1$ are indicated as lines. The point where the lines intersect represents control performance with a perfect model. The cross section along each line exactly represents the plots in Figure 3.2 and Figure 3.3, respectively. As discussed previously, model mismatch in time constant in the direction of $\tau_1<\tilde{\tau}_1$ leads to oscillatory behavior; however, one can see that this oscillation does not occur if there is model mismatch in gain $K<\tilde{K}$ at the same time. As expected, with model mismatch in gain $K>\tilde{K}$ the problem gets worse. The fact that model mismatch in inherently different model parameters can cancel out or amplify the overall effects on control performance shows that it is important to evaluate all dimensions of model mismatch during calculation of optimal controller and state update tuning. It also

becomes apparent that the best control performance is not necessarily achieved with the perfect model. If $K/\tilde{K}=2$ and $\tau_1/\tilde{\tau}_1=1.5$, the control performance is 0.0545 instead of 0.1226 at $K/\tilde{K}=1$ and $\tau_1/\tilde{\tau}_1=1$, a 56% improvement, assuming that the tuning parameters are optimally calculated by the described optimization formulation.

If the assumed process model and the exact process model mismatch were known, the model used by the controller and observer could obviously be substituted with a perfect model to achieve an even better performance. However, since this research is aimed towards deriving a method that does not depend on an exact process model, the discussed results are achieved by only adjusting the tuning parameters defined in 3.1 and leaving the assumed process model unchanged (as this would be the best guess of an engineer in a plant).

Figure 3.6 depicts the equivalent tuning map for MPC with simplified Kalman filtering. The top left corner of both optimal tuning maps indicates a region of instability; both plots are cut off at IAE=2. It is obvious that this region is approached with a very steep slope. Calculating and plotting 3-dimensional tuning maps allows easy assessment of size, location, and steepness of such unstable region(s). Since such regions should be avoided by all means, constraints for highly penalized slack variables can be added into the optimization problem. When comparing the optimal tuning maps of MPC with the different Kalman filter methods, it becomes apparent that both controllers become unstable in region $\tau_1>\tilde{\tau}_1$, $K<\tilde{K}$. But only the MPC with a general Kalman filter becomes unstable in region $[\tau_1<\tilde{\tau}_1, K>\tilde{K}]$. As suggested previously, the inherent filtering that is found only in the simplified Kalman filter acts as a stabilizing mechanism.

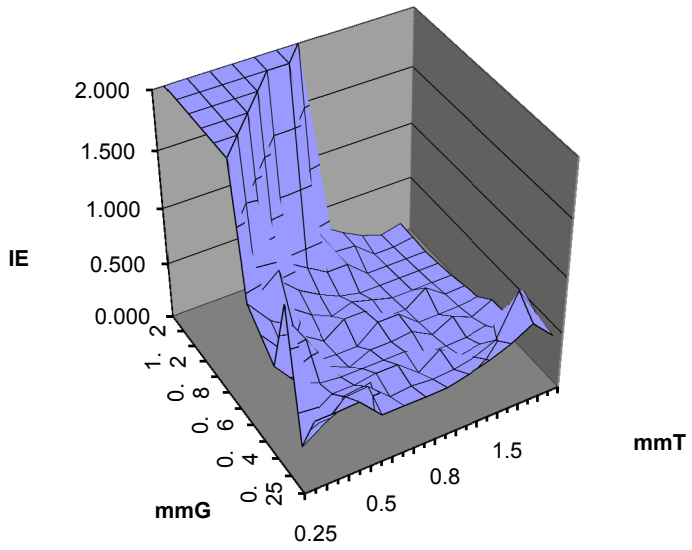

Figure 3.6: Optimal tuning map for model mismatch in K and τ_1 of MPC with simplified Kalman filter state update

The detailed performance and tuning parameter values for the discussed tuning maps are shown in Table 3.4 and Table 3.5.

Process gain K	Process time constant τ_1											
	0.25	0.3	0.4	0.5	0.6	0.7	0.8	1	1.2	1.5	2	2.5
0.25	0.493	0.540	0.711	0.929	1.150	1.326	1.461	1.7294	1.9399	1.9782	1.8572	1.8868
0.3	0.389	0.407	0.505	0.654	0.823	0.997	1.140	1.3334	1.5161	1.7235	1.6796	1.5898
0.4	0.298	0.282	0.302	0.368	0.463	0.567	0.674	0.8897	1.0283	1.1634	1.3821	1.3515
0.5	0.222	0.273	0.225	0.241	0.289	0.354	0.427	0.5737	0.7263	0.873	1.0173	1.1526
0.6	0.293	0.188	0.190	0.186	0.201	0.237	0.285	0.3934	0.4998	0.6621	0.7949	0.9016
0.7	0.362	0.239	0.238	0.158	0.159	0.173	0.202	0.2774	0.3625	0.4833	0.6528	0.7192
0.8	0.429	0.290	0.146	0.168	0.138	0.139	0.152	0.2027	0.2672	0.3666	0.5261	0.6114
1	0.556	0.385	0.205	0.119	0.155	0.112	0.109	0.1226	0.1572	0.2218	0.3308	0.4364
1.2	41.6667	0.4765	0.2619	0.157	0.1004	0.1421	0.0998	0.0903	0.1029	0.142	0.2228	0.297
1.5	33.3333	33.3333	0.6397	0.2132	0.1409	0.2043	0.0686	0.081	0.0721	0.0833	0.1282	0.1835
2	25	25	25	25	0.3322	0.1465	0.1069	0.0617	0.0833	0.0545	0.0632	0.0891
2.5	20	20	20	20	20	0.1935	0.2208	0.0867	0.1187	0.0678	0.0318	0.051

Table 3.4 Optimal tuning map results for model mismatch in K and τ_1 of MPC with general Kalman filter state update. This table corresponds with Figure 3.5.

Process gain K \ Process time constant τ_1	0.25	0.3	0.4	0.5	0.6	0.7	0.8	1	1.2	1.5	2	2.5
0.25	0.622	0.622	0.631	0.431	0.432	0.431	0.435	0.478	0.5267	0.5995	0.7038	0.5233
0.3	0.232	0.356	0.522	0.534	0.359	0.377	0.360	0.3722	0.4077	0.4837	0.8559	0.5504
0.4	1.341	0.266	0.395	0.395	0.424	0.424	0.269	0.4083	0.2733	0.303	0.3617	0.6488
0.5	0.656	0.428	0.341	0.317	0.317	0.317	0.318	0.2152	0.2163	0.2212	0.2563	0.2906
0.6	0.787	0.564	0.595	0.269	0.178	0.265	0.265	0.2848	0.1794	0.2764	0.1945	0.2844
0.7	0.921	0.695	0.571	0.165	0.152	0.152	0.305	0.3056	0.1529	0.1542	0.1571	0.1748
0.8	62.500	0.783	0.935	0.258	0.133	0.268	0.268	0.1333	0.1334	0.268	0.1355	0.1432
1	50.000	50.000	0.616	0.379	0.240	0.152	0.215	0.1615	0.1065	0.1067	0.1077	0.1085
1.2	41.6667	41.6667	41.6667	0.5047	0.3312	0.2191	0.1313	0.0886	0.0886	0.0888	0.089	0.0899
1.5	33.3333	33.3333	33.3333	33.3333	0.4534	0.336	0.2269	0.2287	0.0708	0.0709	0.071	0.0712
2	25	25	25	25	25	25	0.5084	0.2171	0.2024	0.0533	0.0532	0.0809
2.5	20	20	20	20	20	20	20	0.3197	0.2016	0.1033	0.0425	0.0863

Table 3.5 Optimal tuning map results for model mismatch in K and τ_1 of MPC with simplified Kalman filter state update. This table corresponds with Figure 3.6.

3.3 Addressing varying model mismatch

As shown in the previous sections, the optimization problem (3.1) computes the optimal tuning for a particular value of model mismatch, which is one of the original goals of this research. This optimal tuning map is useful for calculating MPC and observer tuning that will ensure optimal control performance. As discussed in the first chapter, industrial users of MPC usually have to manually adjust tuning "knobs" until a desired behavior appears to be reached because such a method was not available before. However, while it is easier to determine the presence of model mismatch than the correct model, one could argue that it may still be difficult to determine the specific amount of model mismatch. Although determining the amount of model mismatch may still be easier than determining the precise model because it requires less process perturbation, the amount of model mismatch may change over time. For these reasons a more feasible and easier to determine measure than model mismatch is the range of

model mismatch. An example of a range of model mismatch in a two dimensional subspace (ignoring the second order time constant as done above) may be written as $K_{actual}=2\pm0.5$ and $T_{actual}=20s\pm5s$ and illustrated in 3-dimensional subspace as depicted in Figure 3.7. Once model mismatch is defined in this way it can be overlaid with the optimal tuning map described above.

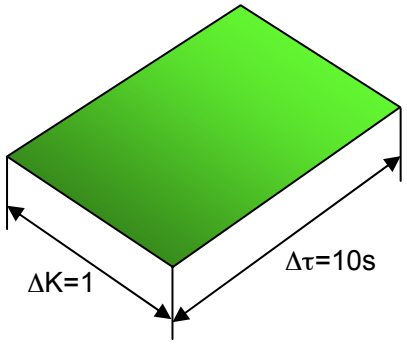

Figure 3.7: Illustration of two-dimensional model mismatch subspace example with $K_{actual}=2\pm0.5$ and $T_{actual}=20s\pm5s$

Such overlay can then be implemented in a software package that displays it to the engineer. This can be very useful to the engineer as he/she commissions the controller because then he/she can assess the worst and best possible control performance as a function of model mismatch for a particular tuning and make manual corrections. An example view of the overlay is shown in Figure 3.8.

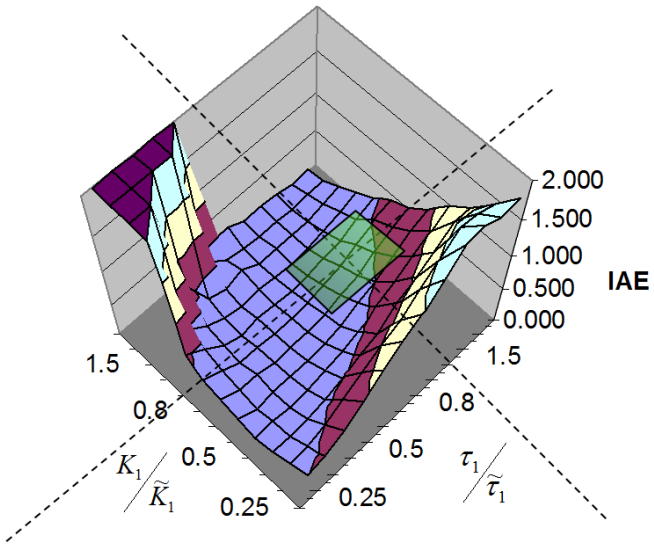

Figure 3.8: Illustration of two-dimensional model mismatch example with K_{actual}=2±0.5 and T_{actual}=20s±5s

In this example, the worst control performance occurs if K_{actual}=1.5 and T_{actual}=25s. However, the IAE at that point is 0.7, which could be considered by the engineer to be acceptable regardless, especially since the likelihood of being within that region is relatively low because only a small surface area overlaps with IAE values above 0.5. If more knowledge about the model mismatch was available (e.g., physical process limitations), then the two-dimensional model mismatch subspace as depicted in Figure 3.7 could be modified to account for likelihood of occurrence. Oval and other shapes are conceivable instead of the depicted rectangle.

3.4 Closed-loop performance improvements from tuning of model parameters

Previous sections showed how the methodology, suggested in this research can determine the most ideal tuning for a given model and model mismatch range. In this work tuning is defined clearly as the MPC controller and observer parameters described in (3.1). However, looking at Figure 3.8, it becomes clear that calculated tuning does not necessarily provide the lowest possible IAE because the center point of the model mismatch subspace is fixed at the assumed "perfect" model. If this subspace, which in this two-dimensional example is represented by a surface, was allowed to move, it could most likely find a lower value for the worst IAE within the surface, thereby increasing the overall control performance.

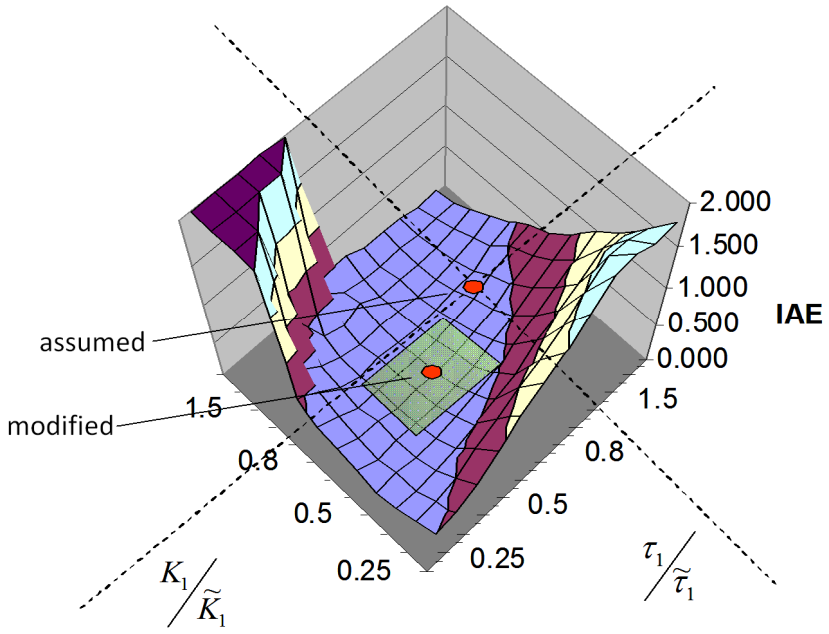

Figure 3.9: Schematic overview of revised optimization method: MPC tuning and design parameters are calculated based on model and model mismatch range

As mentioned previously, this research intentionally does not update the assumed model with knowledge about model mismatch because the result may be as uncertain as the assumed model in the first place. However, as depicted in Figure 3.9, the model used for calculating the MPC control moves may be modified as the described model mismatch surface is relocated within the tuning map. The minimal possible value of the worst IAE within the model mismatch subspace can be found by formulating and solving a second optimization problem.

$$\min_{i} \max_{i\Psi} IAE(\Xi/\hat{\Xi}, \Psi) \quad \text{s.t. } g_{\Psi}(\Gamma) \geq 0 \tag{3.2}$$

where Ψ is the tuning map calculated by iterating (3.1) over arbitrary combinations of model mismatch and $g_{\Psi}(\Gamma)$ are inequality constraints describing the dimensions of tuning map Ψ. It should be noted that no additional process model knowledge is required for this operation because the tuning map is still built based on the model provided by the engineer (the assumed model). The result of the optimization is a modified model that is subsequently used to develop the MPC controller. The tuning map is not recalculated based on the new model because its sole purpose is to minimize IAE within the current tuning map. By adding this operation to the method, the model parameters of the controller are tuned to the ideal values to maximize control performance even further than possible with MPC and observer tuning. In a sense, the model parameters have also become tuning parameters. If there was no model mismatch, one would expect that the control performance of a controller with a modified model would be worse than that of a controller with the original model. However, as discussed above, the chance of that occurring is very small. In a real plant scenario the performance of the original controller may be is worse than that of the

modified controller for the majority of model mismatch scenarios because that is exactly the objective function of the third optimization (3.2). The difference in worst IAE between the assumed and the modified model is usually significant because slopes that lead to instability on one side and low performance on the other side are very steep. Figure 3.10 shows how the schematic overview in Figure 3.1 is modified to use model mismatch range and output a modified model for use in the MPC controller. The need for two optimizations is indicated in the box, and the input is changed from model mismatch to range of model mismatch.

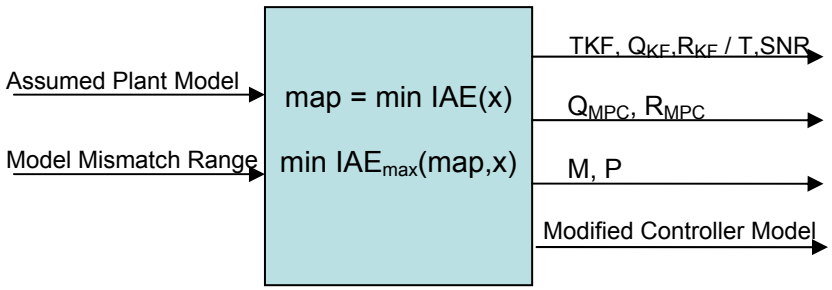

Figure 3.10: Schematic overview of revised method that includes MPC and observer tuning and a modified MPC model-based on model and model mismatch range

Extending the method for optimal tuning from specific model mismatch to a range of model mismatch, as was described in this section, is logical and increases its usefulness dramatically. This new method can be applied to many industrial processes that have inherent process parameter variations that are known, but difficult to measure. The next chapter presents a novel method that uses model mismatch feedback to adapt the presented double optimization method to changing model mismatch.

Chapter 4

Novel method for closed-loop adaptive control

Most methods for adaptive control work by refining or re-creating a process model continuously, or, if triggered by an event such as a change in process value or operator setpoint, spontaneously. After a new model has been obtained, new controller moves or tuning are calculated from it. A few methods adapt the tuning or other controller parameters directly based on process information, without a model, e.g., identification-free algorithms developed by Maršik and Strejc [34]. Both groups of methods rely on process variation that may be introduced by disturbances or setpoint changes. The efficiency, precision and stability of these methods increase proportionally with the amount of process variation.

Common engineering sense suggests that it is easier to determine the statistical amount and variation of model mismatch than it is to determine/create a precise process model. Many methods, such as autocorrelation, have been proposed to determine the amount of model mismatch during closed-loop plant operations [33], [49], [52], [53]. It is extremely difficult to determine a good process model during closed-loop plant operation, however, because the controller objective (to minimize variation of product quality) contradicts the requirement of model identification in order to maximize variation in process outputs.

The method introduced in this chapter is novel because, as a logical continuation of the findings in the previous chapter, it proposes to use the amount of model mismatch to adjust the tuning parameters. While this method, like the one above,

depends on some amount of process variation, it does not need to maximize them in order to derive the tuning that maximizes the control performance.

4.1 Analysis of innovation

The innovation (I_k), also referred to as the residuals or prediction error, is defined as:

$$I_k = (y_k - \hat{y}_k) \tag{4.1}$$

where y_K is the predicted process output and $\hat{y}_k$ is the actual value of the process output. This term is used in the Kalman filter equation (2.8) to calculate the updated state variable. Researchers have proposed many methods that analyze innovation [33], [49]. Often the application of such methods occurs during the commissioning or maintenance phase of a predictive control or soft sensor (e.g., neural network) project. Autocorrelation, for instance, is a method that is frequently used because it allows distinguishing between model error and unmeasured disturbance. Since unmeasured disturbances manifest themselves in the same way as model error to the operator—as the difference between the prediction and the actual value, i.e., a nonzero innovation as seen in (4.1)—they are difficult to distinguish and can only be corrected for by using feedback control.

Autocorrelation of the innovation provides an indication of how much of non-random contributions are not removed from a signal during closed loop control. For a discrete time series of length n $\{y_1, y_2, \ldots y_n\}$ with known mean and variance, an estimate of the autocorrelation can be obtained as:

$$R(k) = \frac{1}{(n-k)\sigma^2} \sum_{t=1}^{n-k} (y_t - \bar{y})(y_{t+k} - \bar{y}) \tag{4.2}$$

where R(k) is the Autocorrelation that lies in the range [-1, 1], is the variance, and is the mean. Since an optimally tuned controller can only remove correlated signals but not disturbances that are perfectly random (i.e. white noise), the proportion of white noise to correlated signal is a good indication of optimality of controller tuning. If the process output of closed loop control has large autocorrelation, then the tuning of the particular controller is not optimal.

Figure 4.1 shows the controlled variable of a PID loop with a relatively large variability (σ=6.8). The presented data was measured on a pressure control loop of an extruder in a chemical plant. Retuning the controller using lambda tuning achieved a 41% reduction in standard deviation. As can be seen in this figure, even with the best possible tuning, the standard deviation is still very large for this particular process (σ=4.0). This makes it very difficult to judge the control performance with the naked eye. Only when the autocorrelation plots are compared can the difference in tuning can be easily seen, as is shown in Figure 4.2.

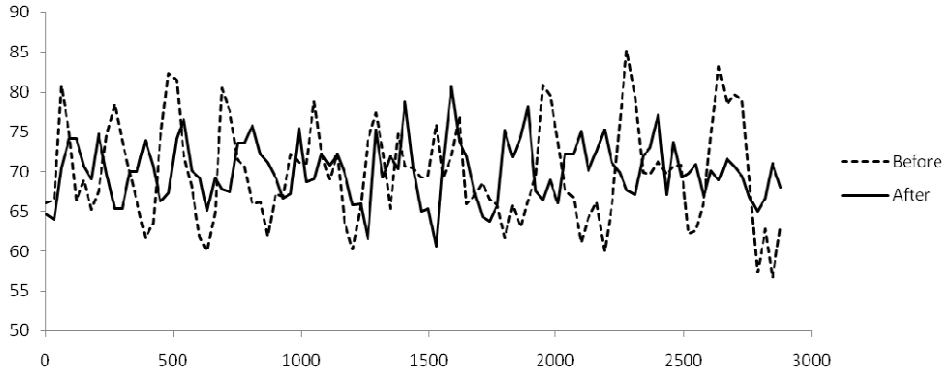

Figure 4.1: Time Series of controlled variable (pressure) of loop with suboptimal and optimal tuning, i.e. before and after lambda tuning; most of the variation is due to process noise, not tuning

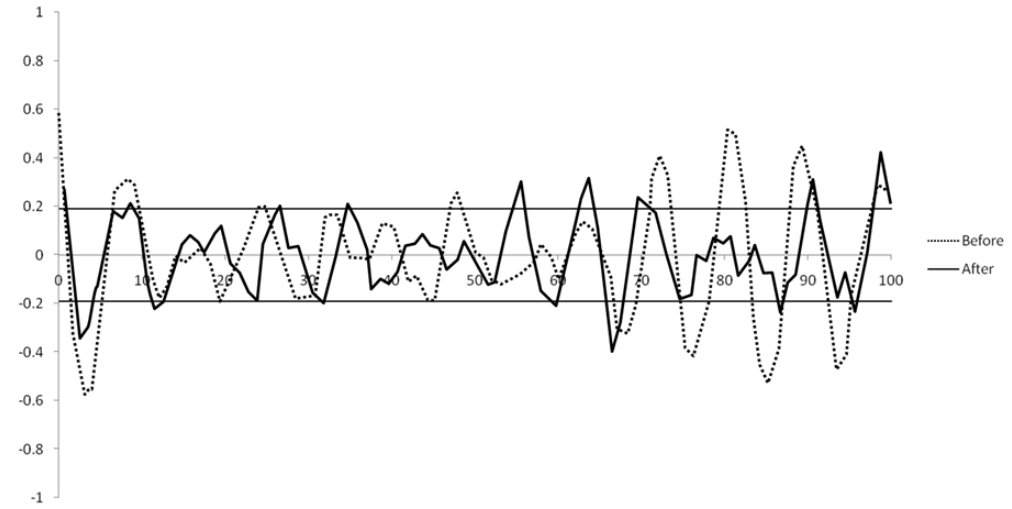

Figure 4.2: Autocorrelation of controlled variable (pressure) of loop with suboptimal and optimal tuning, i.e. before and after lambda tuning; most of the autocorrelation can be removed with tuning

Such methods are frequently used in the manual process of tuning and retuning loops. If the autocorrelation analysis returns a high amount of model mismatch, the engineer usually knows that his work is not completed and he/she must refine or re-identify the process model before he/she can commission the controller or soft sensor. Plant engineers often use a second criterion to verify performance improvements of tuning. They consider the amplitude spectrum to ensure that the amplitude ratio is acceptable at the most likely operating frequencies. In the example shown above, the low frequency attenuation was reduced by 39% from 3.68 to 2.26.

This chapter proposes an automatic method to adapt the controller tuning to the process by virtue of innovation analysis. However, the manual method described above and the new method proposed in this chapter differ on two accounts:

1. The new method is executed continuously during the controller operation and

2. In the new method the amount of model mismatch is not used to trigger a model improvement, but to retune the controller to optimally account for the new amount of model mismatch.

For model predictive controllers the controller output calculation is derived directly from the process model. Therefore, autocorrelation can be attributed to model mismatch.

Current practice in industrial process control deals with the combination of model mismatch and unmeasured disturbance in innovation in a very basic way. DMC controllers, for instance, assume a certain fraction of the innovation to be contributed by unmeasured disturbances. The value can be configured and is usually between 0.5 and 0.8 [27], [32]. However, many researchers have shown this is very inefficient and may lead to significant performance reduction and instability in non-minimum phase systems. A more sophisticated way of selective control action for model mismatch and unmeasured disturbances is the application of the Kalman filter. The Kalman filter is formulated to specifically account for unmeasured disturbances that have a known characteristic. However, this characteristic (covariance), as well as the process model, is assumed to be known exactly. It is also not adaptive with respect to changes in the fraction of unmeasured disturbances in the innovation because the fraction of model mismatch in the innovation is assumed to be zero (since the model is assumed to be known and constant). Han and Lin proposed a method of tuning the Kalman filter gain based on autocorrelation of innovation [33]. This method tries to minimize the error covariance seen by the Kalman filter by adapting the designed and actual signal-to-noise ratios. This essentially maximizes the filter performance, which means that it not necessarily maximizes the closed loop performance as discussed in previous chapters.

Unfortunately, this adaptive method only calculates the error covariances, and can only be used if the perfect model is already known.

While autocorrelation is an indication of randomness in the signal, the R^2 statistic gives an indication about how much of the variation on is explained by the model:

$$R^2 = 1 - \frac{1}{\sigma^2}\sum_{t=1}^{k}(y_t - \bar{y})^2 \tag{4.3}$$

where R^2 is the R-squared value that lies in the range [0..1], σ^2 is the variance, and $\bar{y}$ is the mean. As with autocorrelation, engineers use rules of thumb to determine how much unaccounted variation is acceptable before a model has to be modified. R^2 values above 0.8 are usually considered satisfactory. A value of 0.8 means that 20% of the variation is unaccounted for by the model based prediction, which is very good if achieved in a real plant scenario. However, if the model is over-parameterized, R^2>0.8 may be misleading and a t-test is usually run to ensure the statistical significance of each model parameter. The model can then be reduced by eliminating parameters that turn out to be not statistically significant. The next section discusses how R^2 may be used to infer the approximate fraction of model mismatch versus unmeasured disturbance that contributes to the prediction error.

4.2 Continuous model and tuning adaptation

Gain scheduling methods are very popular in industrial plants for processes with changing process parameters. Such methods may schedule controller tuning as shown by Maršik et al. [34], [35] or the process model and the tuning as shown by Thiele, et

al. [36] and Wojsznis et al. [37]. As long as the process parameters change deterministically, satisfactory results can be achieved with such methods.

In industrial plants many feedstock and equipment properties are constantly changing. Examples include change in BTU properties of fuel and change in concentration of reagent. Other examples are discussed in the following section. If measurable, the values of these property changes are frequently used in one of two different ways:

1. By a feedforward strategy to directly reduce variation or
2. By a gain scheduling strategy to counteract modeling error and indirectly cancel the impact.

Figure 4.3 shows how the optimal tuning method described in Chapter 3 can be combined with an existing MPC controller and observer (indicated as "KF") to create a gain scheduling controller.

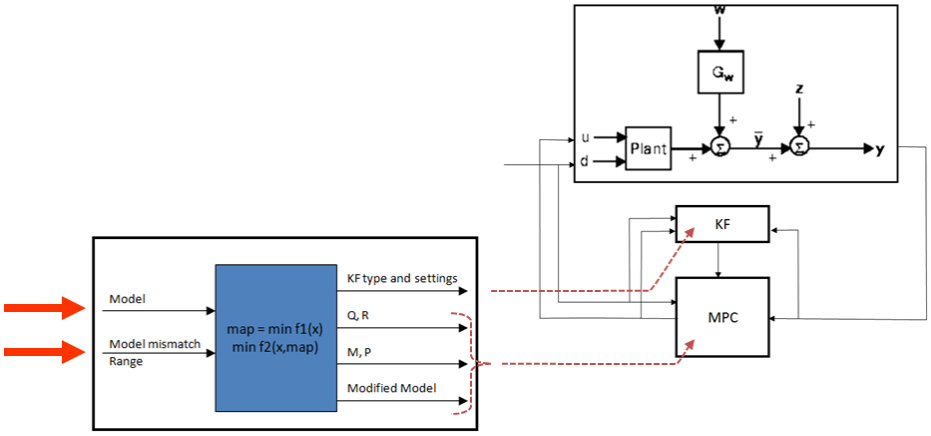

Figure 4.3: Application of optimal tuning method to MPC for manual tuning

If one or more process parameters of the model have changed and are known, they can be updated in the "optimal MPC tuner", which immediately retunes the controller and Kalman filter to counteract negative downstream effects. The update can

be done manually by a plant engineer or automatically based on a process state change. The latter is comparable to model based gain scheduling of a PID controller. The interactions, indicated by dotted lines, show that the information flow to the controller is strictly one-way. Assumed model mismatch range may be entered in the "optimal MPC tuner" by the designing engineer or left at default. As with any gain scheduling controller, the state variable can be 'hardwired' to a specific measurement if available, or estimated by a separate property estimator. Neural networks are frequently used estimators in the industry when process properties can be inferred from process measurements. External property estimators like neural networks or dynamic linear estimators [11] can also be used to estimate key model parameters directly. Such configuration is shown in Figure 4.4. This method closes the loop on model parameters, and can therefore be considered adaptive. However, if we want to maintain the main assumption of this dissertation—that the model is not known—this method would not be feasible. Furthermore, most methods that identify process parameters usually require significant setpoint changes or process perturbation.

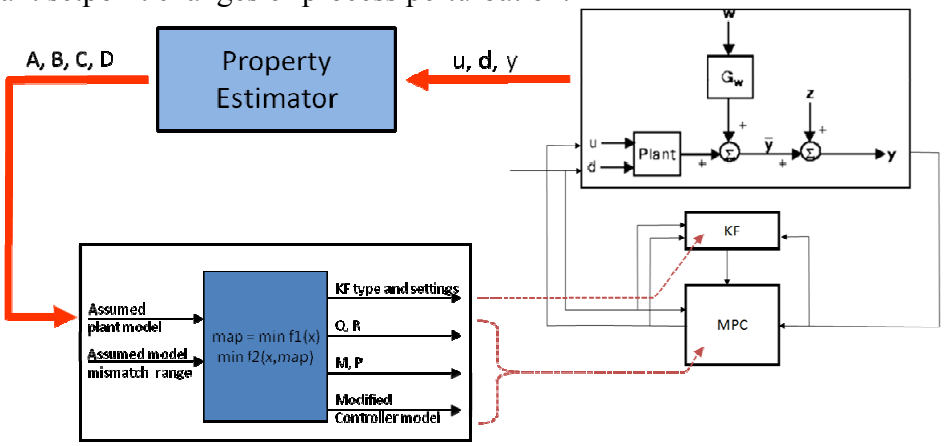

Figure 4.4: Application of optimal tuning method to MPC for adaptive tuning with external property estimator (e.g. Neural Network)

Estimators are usually connected to many variables, including those that are not measured or manipulated by the control loop (shown in Figure 4.4). If the process parameters change non-deterministically or the related state parameters cannot be measured, adaptive tuning methods must be used in lieu of the model based adaptive control discussed above. Examples include recursive model identification and model switching [12], [13]. Figure 4.5 shows an overview of the most common techniques used for adaptive control.

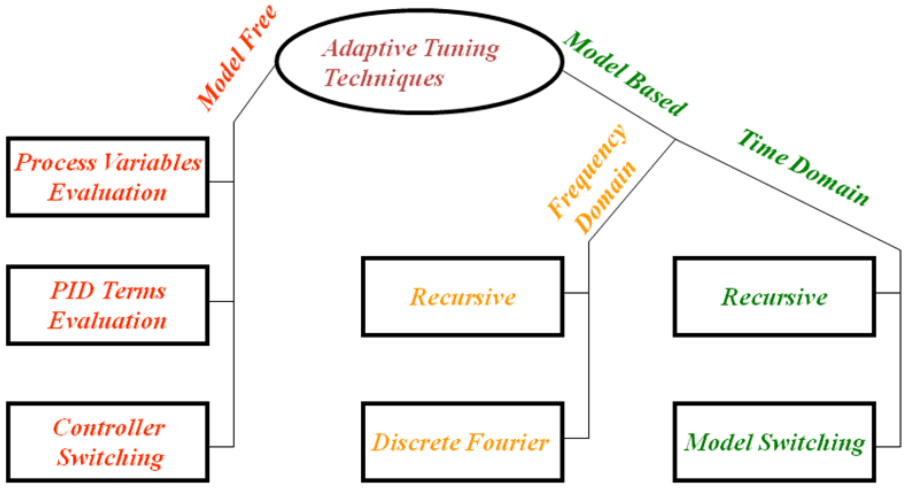

Figure 4.5: Adaptive control techniques

Since model based approaches are intentionally not considered in this research, a large section of adaptive tuning techniques will not be considered. As can be seen in Figure 4.5, model free approaches generally necessitate the modification of the control formulation. The optimal MPC tuning method presented in Chapter 3 is similar in that it only adjusts the MPC tuning parameters. The following analysis will focus on developing a new method that estimates model mismatch without the need for model identification.

Figure 4.6 shows a simple modification to the gain scheduling controller from Figure 4.3 that enables adaptive control. The innovation, which is calculated in the observer, is analyzed and processed by a method (shown as "innovation analysis") that will be discussed below. This "innovation analysis" is used to automatically update the model mismatch range; all or part of the model mismatch range can be updated in this way. Similarly, all or part of the parameters for which range is considered may be modified, depending on how comprehensive the innovation method is with respect to the number of model parameters. In other words, sometimes the innovation may only allow conclusions to be made about a subset of the parameters in the actual model. If that is the case, the range for the unknown parameters must be set conservatively to include all model mismatch scenarios. The output of the optimal MPC tuner functions in the same way as described in the on-demand update cases above, but closes the adaptation loop in this scenario. The uniqueness of this adaptive control approach is that the originally assumed model is never modified by the mechanism, preventing runaway process identification, increasing robustness, and simplifying gain scheduling. If desired, the assumed process model can be updated manually at any time without having to stop or reset the adaptation. This is another advantage over current state of the art adaptive methods. A trigger for manual or automatic model update can be easily derived from the value of innovation or model mismatch range (i.e. either from the input or the output of the "innovation analysis" component).

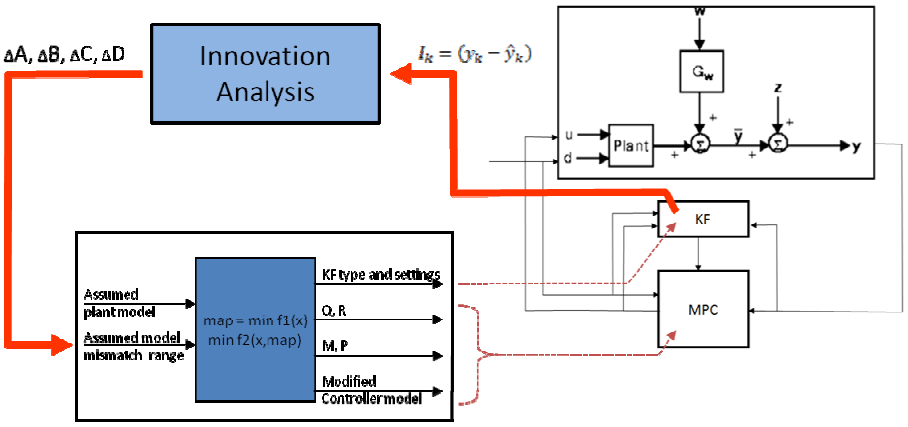

Figure 4.6: Application of optimal tuning method to MPC for adaptive tuning with state estimator (without need for property estimation)

Badwe et al. [49] propose a method for distinguishing between different root causes for innovation. This methodology gives an indication of the differences in magnitude between disturbances and model mismatch. Frequency domain characteristic of the designed sensitivity S_d and the relative sensitivity S_Δ are calculated and compared to empirically derived thresholds. Additionally, stability measures from the small gain theorem [50] are used to rule out tuning problems.

The relative sensitivity is the ratio between the designed and the actual sensitivity.

$$S_a(k) = S_\Delta S_d(k) \tag{4.4}$$

Badwe et al. also show that it relative sens itivity can be used to calculate the error (4.5), controller output (4.6) and spectral density ratios (4.7) of designed versus actual loops:

$$e_a(k) = S_\Delta e_d(k) \tag{4.5}$$

$$u_a(k) = S_\Delta u_d(k) \tag{4.6}$$

$$\phi_{e_a}(j\omega) = |S_\Delta(j\omega)|^2 \phi_{e_d}(j\omega) \tag{4.7}$$

If the frequency response $|S_\Delta(j\omega)|^2$ is known, the condition for marginal stability $\|S_\Delta\|_\infty > 1$ can be used to detect model mismatch. Unfortunately, this calculation requires the identification of an updated process model. Since the initial goal of this dissertation is to develop a controller tuning method that compensates for model error without the need for model identification, the two cited methods cannot be utilized. Methods that calculate model mismatch based on the comparison between two models are usually theoretical methods and not realistic in a plant environment. An "actual" plant model is only hypothetical since it cannot be more accurate than the model that is already assumed to be the process model. Other methods have been proposed to determine performance degradation due to model mismatch or other causes. The cross-correlation test by Stanfelj et al. [51] uses the cross-correlation between setpoint variations and internal model control error to discriminate between process model error and disturbance model error. However, it requires SP perturbations to the system and is designed for single loop controllers.

The autocorrelation of the innovation method described in the previous section provides a model-free and perturbation-less alternative. As discussed above, autocorrelation analysis can determine whether a loop's control performance can be improved or not without the need to identify a process model. Kesavan & Lee [52] argue the same point but from a different perspective. In their work on diagnostic tools, they compare autocorrelation of innovation before and after detuning a model based controller in order to determine whether model mismatch is present or not. If model mismatch is present, then two different controller tuning settings will result in different

autocorrelations. Currently this is a manual process. Such two process control scenarios can be created by either running with two sets of controller tunings for two periods of time, or by turning control off for some amount of time (open loop measurement). Automating this method fully requires solving many challenges that are not part of this dissertation research. Besides, as is the case with all well-defined safety nets, the controller operation must be very transparent to the user. Even with safety nets fully automated, the calculation of model mismatch using this particular method may still be too disruptive and impractical for some types of processes or plants. Further research like that discussed in section 2.4 needs to be done on this method of two tuning scenarios to understand the effects of different types of model parameters and their interactions. Furthermore, it has yet to be determined how much the tuning settings have to differ in order to see significant differences and what the impact of the number of samples (test duration) in each tuning scenario on the overall confidence is. The following section discusses the results from experimental test runs of a model predictive controller that uses the optimal tuning method described in this chapter (Figure 4.6) using a practical approximation to estimate model mismatch from autocorrelation of innovation.

4.3 Case study: Level loop in distillation column process

To verify the concepts presented in Chapter 3 and Chapter 4, the proposed tuning methods were applied to a binary distillation column. This particular column is a pilot plant of smaller than average scale that is used to separate water and ethanol. The process and instrumentation diagram is shown in Figure 4.7.

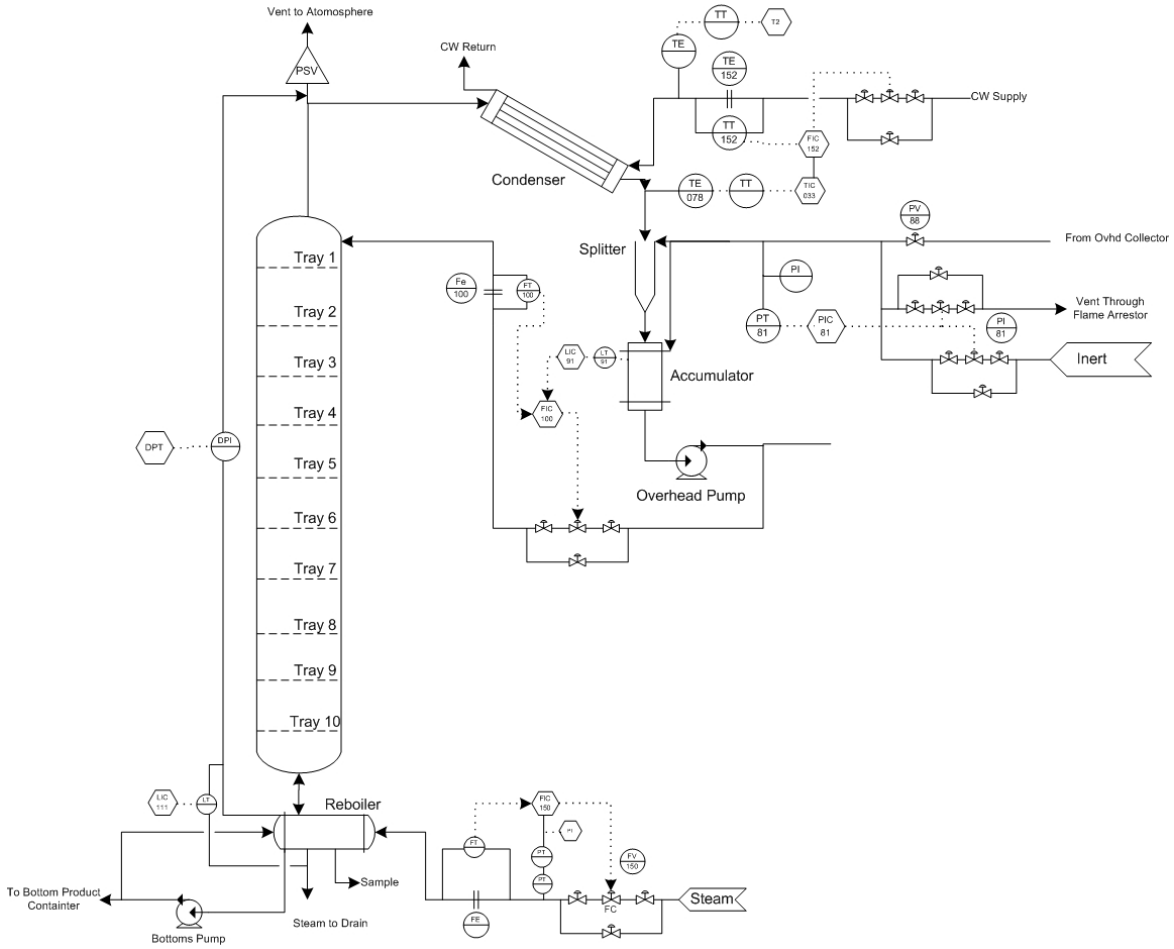

Figure 4.7: P&ID of binary distillation column used for experimental testing of proposed methods

Since the flow out of the accumulator can be measured, a cascade strategy was chosen that allows separating the fast flow and the slow integrating level dynamics. This separation between level control (LIC-091) and flow control (FIC-100) generally increases robustness. However, both controllers must be tuned fairly well for this effect to be realized. This is a challenging task in this plant because the process parameters change as the column energy input varies. Since steam flow is used to control the bottoms temperature in order to control the purity, the process parameters of level and

flow control loops in the accumulator can change significantly during normal operation. Manual step testing was performed to determine the process model at a steam flow of 0.55 kg/min. The step test yielded the following initial model, which will be used as the assumed model for the model based controllers and will provide initial tuning for the PID controllers: $G(s)_{FIC-100} = \frac{0.61}{3s+1}e^{-0.6s}$; $G(s)_{LIC-091} = \frac{-0.9}{218s+1}e^{-17.5s}$. (At steam = 0.4 kg/min: $G(s)_{LIC-091} = \frac{-0.9}{160s+1}e^{-14s}$). Even though these models are FOPDT, they were used for further analysis of the level loop. This approximation of an integrating process with a FOPDT model was done intentionally to guarantee model mismatch during the experiments. Some commonly used tuning rules, such as Ziegler-Nichols, will result in the same PID tuning anyway because they are solely based on ultimate gain, ultimate period and deadtime. Practitioners often try both model types on integrating processes to see which performs better and/or is more robust.

For the experiments the level controller LIC-091, which is normally a PID controller was replaced with MPC control that was executed in MATLAB. This was implemented by leaving the PID controller in manual mode and sending OPC writes from an OPC client on the laptop computer to the outputs of the PID controller in the DeltaV system. Figure 4.8 shows how the open standard OPC protocol is used to read/write access real-time plant data, which is owned by the control system, from an external laptop. The output of LIC-091 is connected to the cascade input of the secondary flow controller FIC-100. Thus writing to LIC-091 indirectly manipulated the flow setpoint of FIC-100 and cascade control behavior, equivalent to the original plant configuration, is retained.

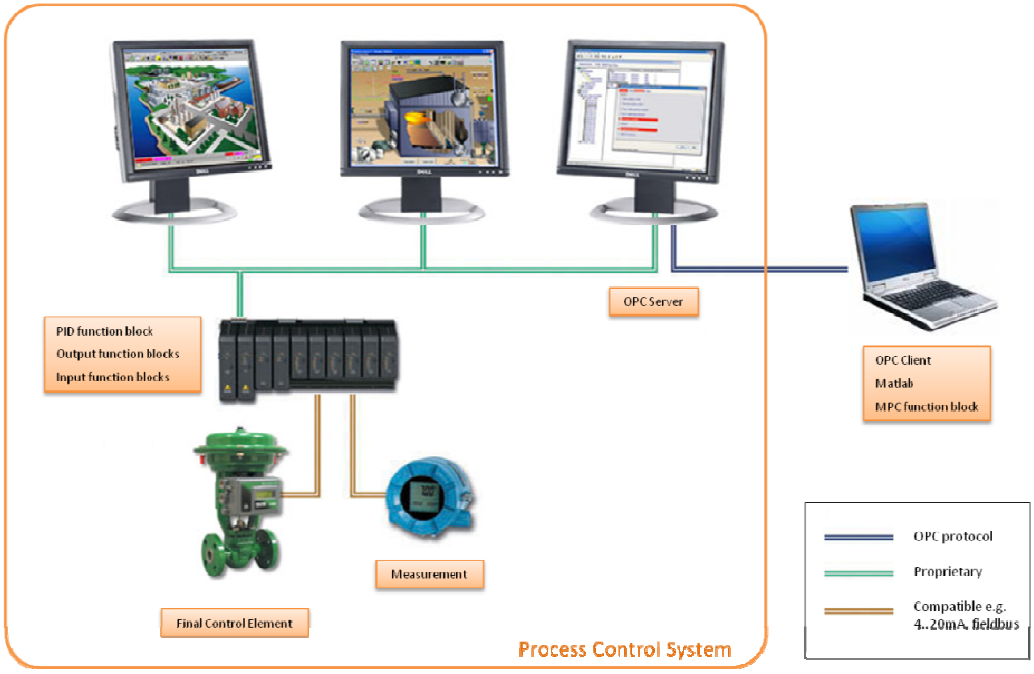

Figure 4.8: Schematic diagram of OPC connection between MATLAB laptop and DeltaV Control system. OPC is an open standard by the OPC foundation [55] for open communication between windows based computers and industrial plant automation.

Three different move penalty tunings were used on the MPC controller (R_{MPC}=50, R_{MPC}=100, R_{MPC}=1000) and the control performance at two different operating points was analyzed (steam flow rate of 0.4kg/min and 0.55 kg/min). Figure 4.9 shows three different runs of the model predictive controller with the three different tuning settings at a steam rate of 0.5kg/min that were recorded sequentially and then overplayed for comparison. A fourth experimental run using the original PI controller was performed to allow comparisons between MPC and PI control, similarly as in previous chapters. The PI tuning (Gain=1.54 and Reset=141.68s) was calculated from ultimate gain, ultimate period and dead time with modified Ziegler-Nichols tuning using DeltaV Tune, a tool for automatic controller tuning. DeltaV Tune was developed by a team at Emerson Process Management lead by the author [56].

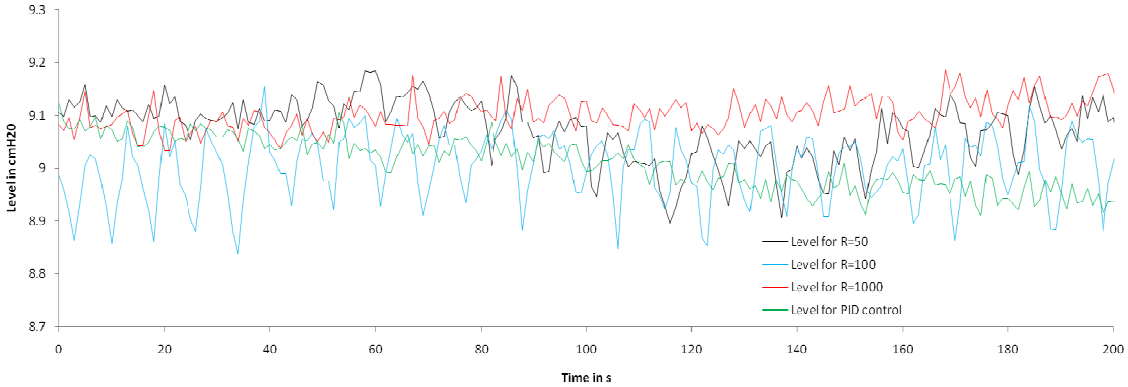

Figure 4.9: Level control at steam rate of 0.55 kg/min for different MPC tuning, $\sigma_{R=50}$=0.057, $\sigma_{R=100}$=0.066, $\sigma_{R=1000}$=0.052, σ_{PID}=0.036

The time domain plot of level control with R_{MPC}=1000 appears to be the most steady. MPC with R_{MPC}=1000 also achieves the lowest standard deviation ($\sigma_{R=1000}$=0.052). In a plant a plot of current control variables is often the main way of looking at the data. However conclusions drawn from a real time trend only may be misleading. Figure 4.10 shows how the controllers react to an unmeasured disturbance.

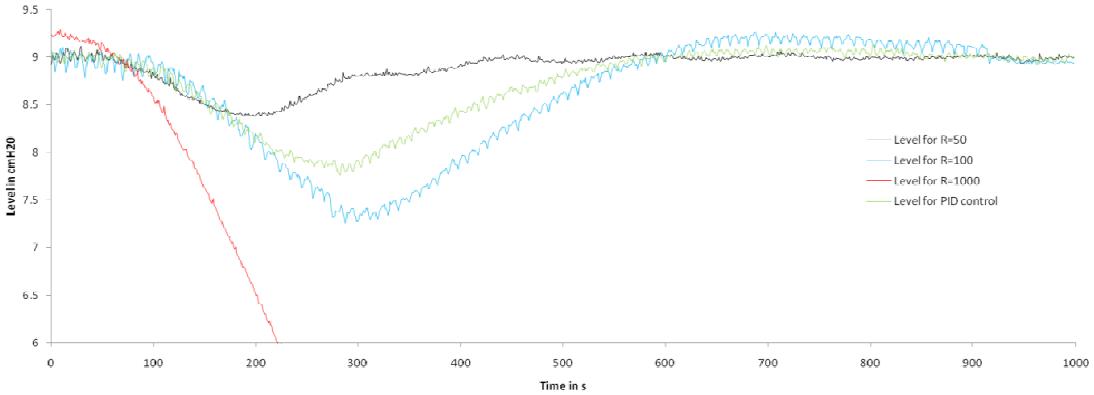

Figure 4.10: Level control after introduction of artificial unmeasured disturbance in steam flow: steam flow loop setpoint was changed from 0.55kg/min to 0.4 kg/min; $IAE_{R=50}$=0.122, $IAE_{R=100}$=0.468, $IAE_{R=1000}$=∞ (control was unsatisfactory and the plant had to stabilized with manual intervention), IAE_{PID}=0.3

The unmeasured disturbance for all experiments was chosen to be a change in steam flow as it poses more than one difficulty for the controller. First, it changes the

amount of condensate that reaches the accumulator which requires the controller to change the accumulator output flow and secondly it changes the reflux time constants, thereby changing the amount of model mismatch. The change in steam flow is also a true input disturbance as described in 2.2. Such process parameter change is common in industrial plants. It is also most commonly unmeasured. Examples from different process industries that have similar impact as the artificial steam flow change in this experiment include:

- Change in BTU rating of fuel (impacts temperature and gain of temperature loop)
- Change in concentration and/or composition of feedstock (impacts column loading, mass balance and gain between energy supply to product purity)
- Fouling of tubes in boiler (changes heat transfer coefficient, therefore changes gain; but also changes required flow for the same heat transfer therefore changes dead time)
- Change of outdoor temperature and or rainstorm (changes temperature but also heat transfer coefficient to atmosphere, therefore changing gain)

As shown in Figure 4.10, while $MPC_{R=50}$ and $MPC_{R=100}$ rejected the unmeasured disturbance reasonably well (with $IAE_{R=50}$=0.122, $IAE_{R=100}$=0.468 respectively), the same experiment could not be performed with $MPC_{R=1000}$ at all because the large move penalty prevented the controller from reacting to the level drop in timely manner which tripped the accumulator pump interlock. Following the pump trip the accumulator filled too quickly for the controller to react.

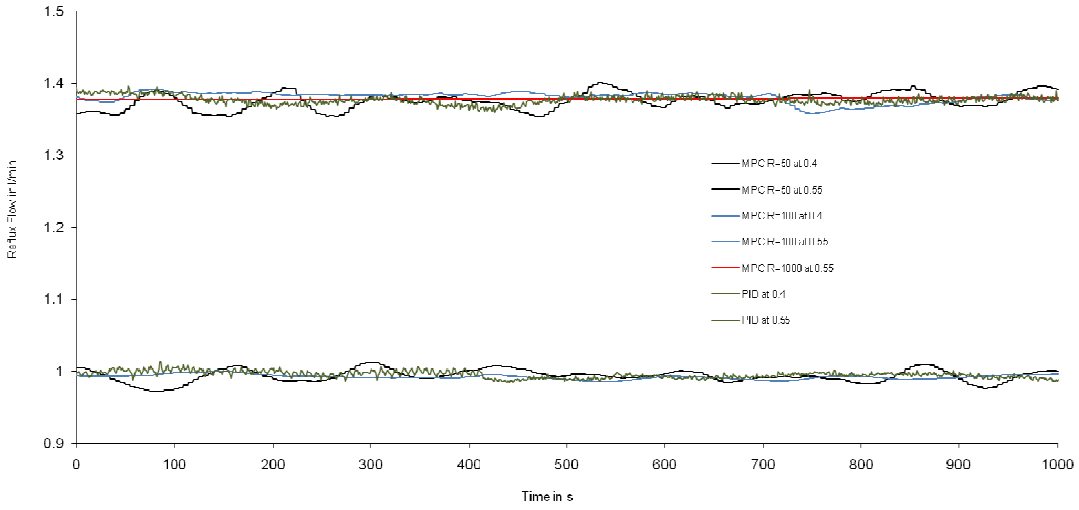

Figure 4.11: Level control at steam rate of 0.4 kg/min for different MPC tuning, $\sigma_{R=50}$=0.053, $\sigma_{R=100}$=0.028, controller with R_{MPC}=1000 did not control plant satisfactory and tripped accumulator pump interlock repeatedly, σ_{PI}=0.032

This example shows that overly cautious detuning of a controller can be potentially disruptive and dangerous. Here a spill (accumulator overflow) would have occurred without manual operator intervention. On the other end of the tuning spectrum the plot of the controller output of $MPC_{R=50}$ in Figure 4.11 suggests that the low penalty on move causes the output moves to be unreasonably large and the controller may be very close to its stability margin. In further experiments this proved to be true because when R was reduced just slightly below 50 or the plant was run in an operating region with higher steam rate than 0.55kg/min the controller became instable (not depicted).

Interestingly, Figure 4.11 also shows that the output of the PID controller is performing quite well with respect to overall standard deviation. However, with the current tuning, i.e. the tuning that is most appropriate for this particular process, this controller appears very noisy. Such small quick control moves may either be absorbed

by the valve hysteresis or cause the valve to wear out quicker than necessary. This effect could be reduced by running the PID controller at a slower sampling rate.

When the plant was operating at 0.4kg/min of steam (Figure 4.12), MPC with the same tuning that was used for 55kg/min of steam (Figure 4.9) showed lower standard deviation ($\sigma_{R=50}$=0.053, $\sigma_{R=100}$=0.028). MPC with R_{MPC}=1000 is not shown in Figure 4.12 because it did not control plant satisfactory and tripped accumulator pump interlock repeatedly, as mentioned above.

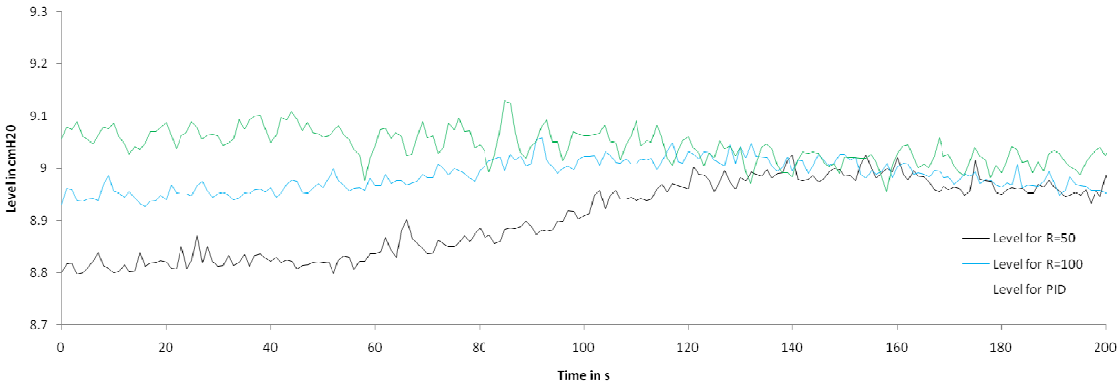

Figure 4.12: Level control at steam rate of 0.4 kg/min for different MPC tuning, $\sigma_{R=50}$=0.053, $\sigma_{R=100}$=0.028, controller with R_{MPC}=1000 did not control plant satisfactory and tripped accumulator pump interlock repeatedly, σ_{PI}=0.032

In summary, the experimental run of the distillation column showed that for MPC tuning parameter R_{MPC} values of 50 and 100 are suitable for model mismatch that spans across the desired operating regions (here from 0.4kg/min to 0.55kg/min of steam). Tuning below 50 and above 200 (tested but not depicted) is unsuitable. As suggested in Chapter 3, specifically section 3.3, a tuning could be found that results in ideal feedback control performance for the varying amount of model mismatch. This was possible experimentally, even without new knowledge about the model. No model re-identification or model update was performed. The assumed model was used throughout the experiment even though it was known to be significantly wrong. Not

only the model parameters were most likely incorrect but also the model form did not match the underlying process characteristics of an integrating process (level loop). An incorrect model formulation (FOPDT) was intentionally selected to ensure that model mismatch was present during the experiment because it is not possible to determine the exact amount of model mismatch that was present during the experiment. A summary of the control performance of four different controllers is shown in Table 4.1

	PI	$MPC_{R=50}$	$MPC_{R=100}$	$MPC_{R=1000}$
$б_{steam=0.55kg/min}$	0.036	0.057	0.066	0.052
$б_{steam=0.4kg/min}$	0.032	0.053	0.028	not tested
$IAE_{steam\ change\ 0.55\rightarrow 0.4\ kg/min}$	0.302	0.122	0.468	∞
$IAE_{steam\ change\ 0.4\rightarrow 0.55\ kg/min}$	0.281	0.136	0.424	not tested

Table 4.1 Summary of control performance of experimental data for three different MPC controllers, all controllers are tuning based on the same model assumptions, PI added for comparison

4.4 Experimental Analysis of Autocorrelation

The question that is still unanswered is: "How does one know while running the plant at one operating point how changing to another operating point will impact the performance for a certain tuning?" Or for the discussed experiment: "How could one know that, although penalty tuning of R=1000 shows the best control at 0.55kg/min, it will be insufficient for larger disturbances, potentially causing a plant shutdown, *before* steam changes from 0.55kg/min to 0.4kg/min?" Manually passing through all regions and determining the one tuning that will work in all regions is feasible and does not require model identification, as described above. However it may create a certain

amount of wasted or below quality product, may have to be redone if the model changes and is only possible if the parameter that impacts the model mismatch is known and can be manipulated (as steam in this example).

As was discussed in section 4.2, other research suggests the use of autocorrelation of the prediction error to determine how well a controller is matched to a plant and what the degree and type of model mismatch are. Figure 4.13 shows a comparison of the autocorrelation of prediction error in MPC at the three different tuning settings at 0.55kg/min of steam.

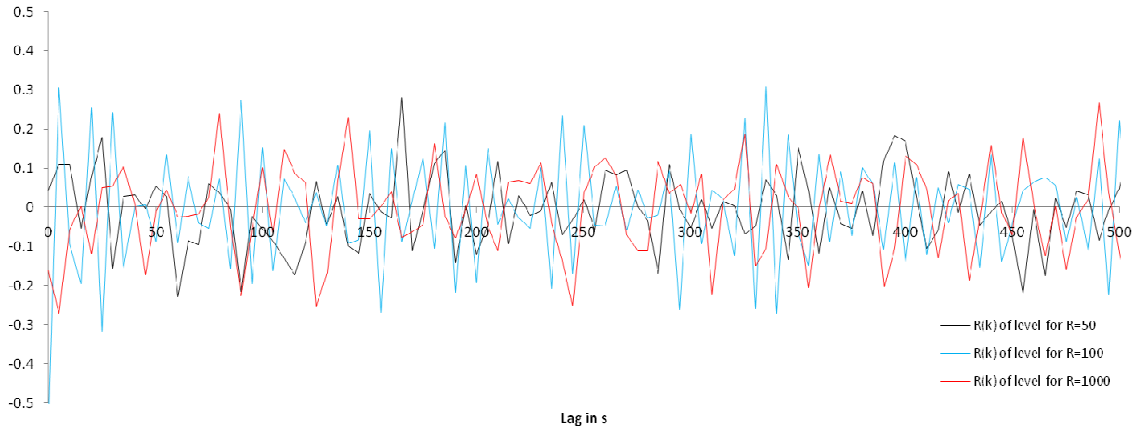

Figure 4.13: Autocorrelation of prediction error in MPC at the three different tuning settings at steam rate of 0.55 kg/min

It appears that no conclusive statement can be made about which plot exhibits better or worse qualities. The amplitudes are different, but considering the lag axis there is no indication that any one of the plots is significantly more self correlated for a given lag time, even though the three controller tunings exhibit significantly different feedback performance in terms of IAE, as was shown in Figure 4.10.

The plots of autocorrelation of prediction error for steady operation at 0.4kg/min, that are shown in Figure 4.14, show the same dilemma.

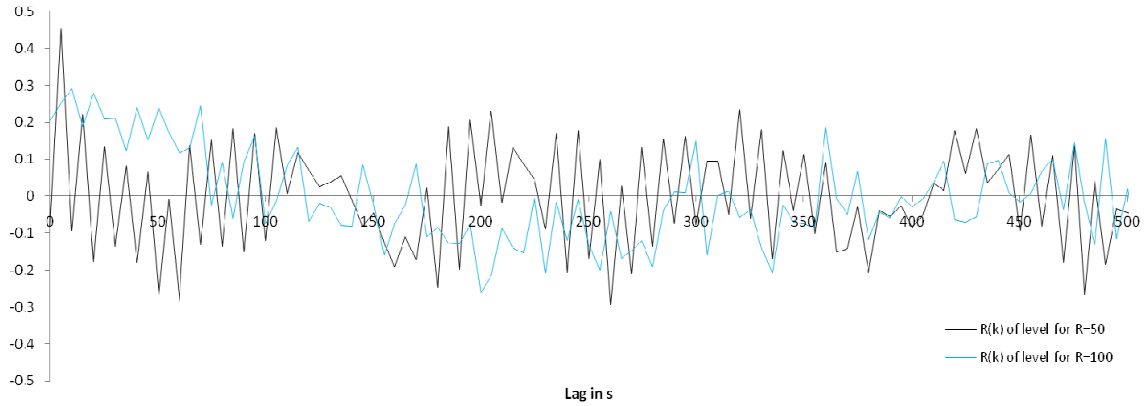

Figure 4.14: Autocorrelation of prediction error in MPC at the three different tuning settings at steam rate of 0.4 kg/min

This supports the statement made in 2.3 that the best state update (lowest prediction error) does not necessarily lead to the best feedback control performance. Here MPC with R_{MPC}=50 shows the lowest prediction error but the worst integrated absolute error and MPC with R_{MPC}=1000 shows the best IAE with the worst prediction error. Differences between the three controller settings can only be observed when the autocorrelation of prediction error during a large unmeasured disturbance change is recorded (Figure 4.15).

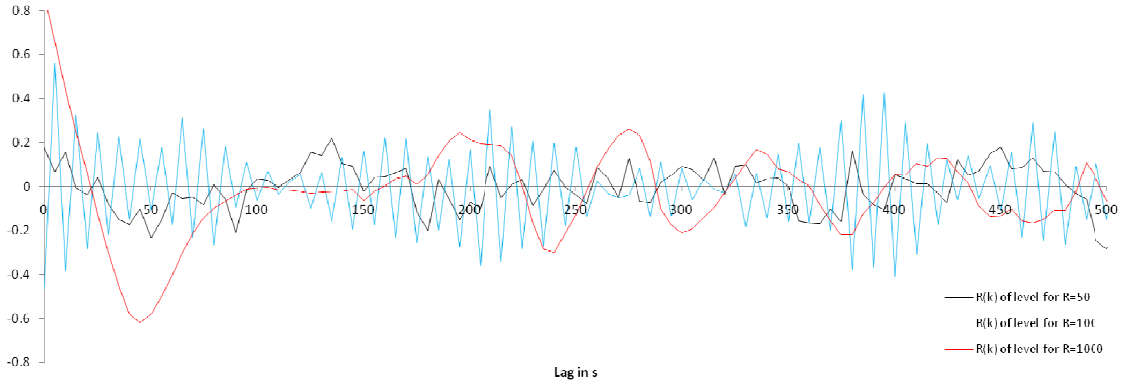

Figure 4.15: Autocorrelation of prediction error in MPC at the three different tuning settings during rejection of unmeasured disturbance (steam rate changes from 0.55kg/min to 0.4 kg/min)

The large change in the control error brings out a noticeable difference between the autocorrelation plots, which is otherwise washed out by noise. Unfortunately at that point is too late to find out that the controller was tuned badly because that information was supposed to be used to retune the controller before a large disturbance occurs. Looking closely through the procedures in papers that suggestions autocorrelation as a criteria for determining model mismatch (e.g. [49] discussed in 4.2) it becomes apparent that such methods also use process excitations to get conclusive comparisons of autocorrelations. Even though some methods may be considered "not intrusive" because they wait for unmeasured disturbance changes rather than injecting pulses that perturb the process, they will only function during steady control, like the steady operation at 0.55kg/min of steam in this experiment.

Use of the autocorrelation of the prediction error for control performance assessment during periods of steady operation turns out to be not very useful due to reasons given above. However, the experimental data shows that looking at the autocorrelation of control error instead could be a much better alternative. Figure 4.16 shows the autocorrelation of the controlled variable which is equivalent to the autocorrelation of the control error for pure feedback control, i.e. with constant setpoint. MPC with the same three tuning settings are shown and the original PI is added for reference again.

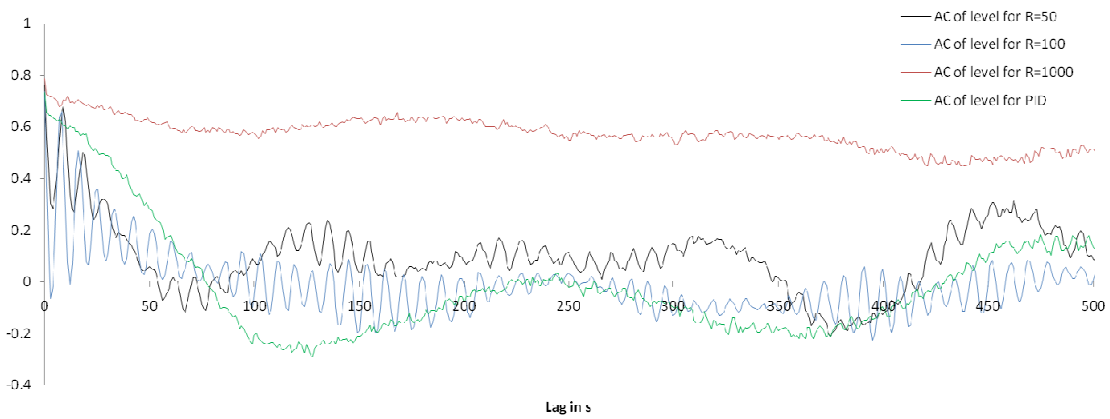

Figure 4.16: Autocorrelation of level for MPC at the three different tuning settings at steam rate of 0.55 kg/min, autocorrelation of level for well tuned PI control is added for comparison – $MPC_{R=1000}$ stands out significantly, easy to determine tuning problems

It can be seen that the autocorrelation of the controlled variable of MPC with R_{MPC}=1000 is vastly different from all other controllers. At steady operation and without any significant unmeasured disturbance this tuning can easily be identified as bad because it shows significantly higher self correlation for all values of lag time. The most obvious distinguishing criterion of this curve is that stays on only one side of the abscissa and never crosses zero.

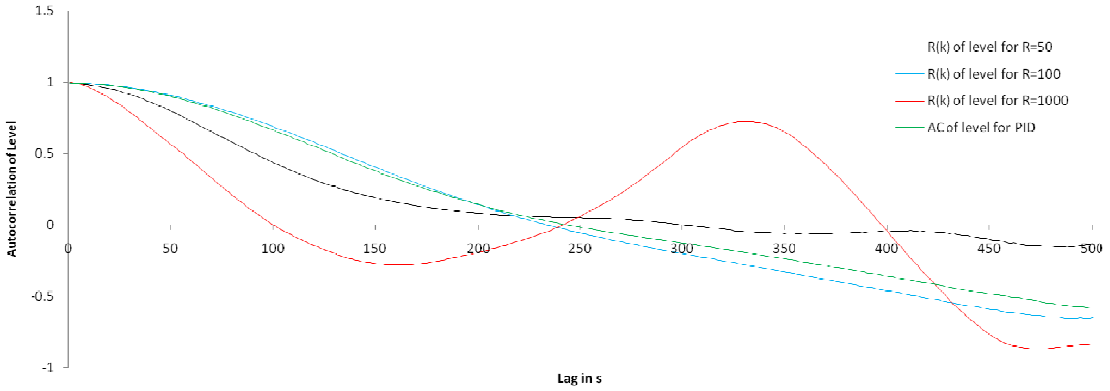

Figure 4.17: Autocorrelation of level for MPC at the three different tuning settings during rejection of unmeasured disturbance (steam rate changes from 0.55kg/min to 0.4 kg/min), autocorrelation of level for well tuned PI control is added for comparison – no

controller stands out significantly, this analysis for tuning problems may not be very conclusive

Figure 4.17 shows the same calculations during the artificially introduced disturbance to the accumulator level that resulted from changing the steam from 0.55kg/min to 0.4kg/min. Although a difference in autocorrelation could be argued, it is not nearly as distinct as without the unmeasured disturbance (Figure 4.16). However this is does not present a significant problem because an automatic method can easily detect a disturbance and switch from analyzing autocorrelation of control error to autocorrelation of prediction error.

A simplified qualitative summary of the autocorrelation analysis discussed in this section is shown in Table 4.2. Experimental data for the four different controllers is displayed and the criteria that can be used to distinguish autocorrelation are highlighted.

	PI	$MPC_{R=50}$	$MPC_{R=100}$	$MPC_{R=1000}$
$R_I(k)_{steam=0.55kg/min}$	n/a	small	small	small
$R_I(k)_{steam=0.4kg/min}$	n/a	small	small	not tested
$R_I(k)_{steam\ change\ 0.55\rightarrow 0.4\ kg/min}$	n/a	***small***	***medium***	***large***
$R_y(k)_{steam=0.55kg/min}$	***small***	***small***	***medium***	***large***
$R_y(k)_{steam=0.4kg/min}$	small	medium	medium	not tested
$R_y(k)_{steam\ change\ 0.55\rightarrow 0.4\ kg/min}$	large	large	large	large

Table 4.2 Summary of qualitative estimates of autocorrelation experimental data for three different MPC controllers, all controllers are tuning based on the same model assumptions, $R_I(k)$ – autocorrelation of innovation or prediction error, $R_y(k)$ – autocorrelation of controlled variable, PI controller added for comparison, criteria that can be used to distinguish autocorrelation are highlighted

In summary, closed loop adaptive control of tuning parameters can be accomplished by a method that analyzes autocorrelation such as the one presented in

this section. What was learned from the plant experiment is that it makes a big difference whether autocorrelation is calculated from prediction error or control error. Autocorrelation of prediction error is only conclusive during a setpoint change or rejection of an unmeasured disturbance while autocorrelation of control error is most useful during steady operation. It was also shown that it is most useful (but also most challenging) to adjust tuning to the process characteristics before an unmeasured disturbance occurs and not during or after, as is done by most current state of the art adaptive tuning methods. Methods that try to re-identify the process model usually rely on process changes and cannot detect model changes during steady operation. Such changes may be caused by disturbance or setpoint changes and must be large enough to be distinguishable from the noise band. Figure 4.18 shows the final modification that was is required in the adaptive strategy shown in Figure 4.6 in order to detect model mismatch before it can cause problems during unmeasured disturbances. The automatic method, shown as "Model Mismatch Analysis" in the schematic diagram in Figure 4.18 should include analysis of innovation (i.e. prediction error) and analysis of control error in the conditional manner described above. The result of this analysis is how well the current tuning is suited for the current process. Any worsened autocorrelation function (as compared to expected or previous autocorrelation functions) for a given tuning must be the result of increased model mismatch and can be accounted for by new tuning.

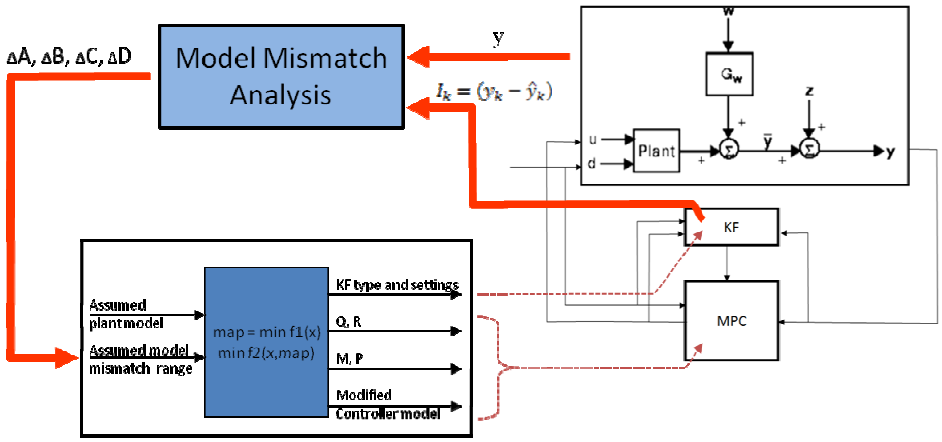

Figure 4.18: Application of optimal tuning method to MPC for adaptive tuning with state estimator (without need for property estimation)

The experimental runs and examples of the use of autocorrelation as a means to feedback information about model mismatch into the control algorithm are by no means comprehensive and merely serve as an example of the concept presented in this chapter. A wider variety of process and model types may be researched in future research. Also the future work could be done to quantify and normalize the amount of model mismatch in a better metric than the shown autocorrelation plots. Furthermore results may vary depending on which type of model mismatch one encounters. In general, the tuning that the optimal tuning method calculates for a wider range of model mismatch will be more conservative than the tuning that results from a narrower range. In other words, the automated tuning method presented above detunes the controller to prevent oscillations and instability, acting like an automatic safety net that kicks in whenever needed. This automatic approach is obviously more desirable than proactively detuning the controller to be safe, a common practice in industry, because it automatically applies faster tuning when the model mismatch is small.

The distillation example experiment discussed above is meant to prove the general concept of adaptive control based on model mismatch. It is certainly not

completed research on adaptive model predictive control. As with any model based controller, an initial process model must be assumed and this method does not fall in the "Model Free" category of adaptive controllers. However, this method shows that a model based controller can be adapted to model mismatch without model identification. As was discussed in Chapter 1, model identification, particularly closed loop model identification, has proven to be either intrusive or unreliable in industrial process applications because the objectives of control and identification contradict each other.

Chapter 5

Novel method to improve MPC control performance for model mismatch

Previous chapters showed how optimal tuning can be derived from statistical knowledge about model mismatch. This chapter discusses how tuning of feedback—the ability to reject load changes, such as unmeasured disturbances—can be improved with empirical tuning. As discussed in previous chapters, the feedback performance of model-based controllers can be tuned to be ideal and as good as the setpoint change performance if the model is perfectly known. When there is model mismatch, however, the tuning is chosen more conservatively by the engineer. The method described in this chapter offers an alternative to detuning the controller. If the controller reacts to an increasing accumulation of prediction error, i.e. slope, it can account for unmeasured disturbances quicker than standard MPC. As the MPC control equations (1.1) to (1.3) show, controller outputs only depend on previous inputs, outputs and setpoints, not on previous control errors. Previous control errors should be proportionally related to previous setpoints. This proportional relationship only holds true, however, if there are no unmeasured disturbances. If unmeasured disturbances occurred in the past, then it is important to consider them in the control equation. The method presented in this chapter addresses this issue and uses this information to drive the controller output directly, thereby improving feedback control performance. This chapter also discusses how a method that improves this aspect of control can be implemented to avoid adverse

effects on setpoint change performance and overall performance in the absence of model mismatch.

The author's hands-on experience with a large chemical plant site in Alabama in 2000 provided special motivation for the research contained in this chapter. Two challenging processes to prove the concept of embedded MPC control in DeltaV [43], a modern process control system developed by Emerson Process management (formerly Fisher-Rosemount). In [17] the author discusses benefits and challenges of embedded MPC, which allows model predictive control in an embedded controller hardware unit in the field and has many advantages over third party solutions, such as redundancy, fast sample periods, easy and seamless configuration, and commissioning without disruption to the running process. The author now holds a patent of this concept [18]. In the beginning of the research project the author asked the engineers at the test plant to select two of their most challenging processes as candidates for conversion from PID to MPC strategies. One of the challenging processes was a decoupling application which was successfully implemented by the plant engineers and lead to outstanding performance improvements. The other application was a single input process that depended heavily on feedback control because the process parameters were drifting significantly and many unmeasurable disturbances from upstream and downstream units were impacting it. Five control experts, two of whom are members of the Control "Process Automation Hall of Fame", tried at first to improve the original PID feedback control performance by using MPC and later to at least match MPC to what was observed with PID for over five days without success. Figure 5.1 shows the best MPC disturbance rejection performance that could be achieved compared to the disturbance

rejection of PID control which was used to control this particular process for more than 50 years.

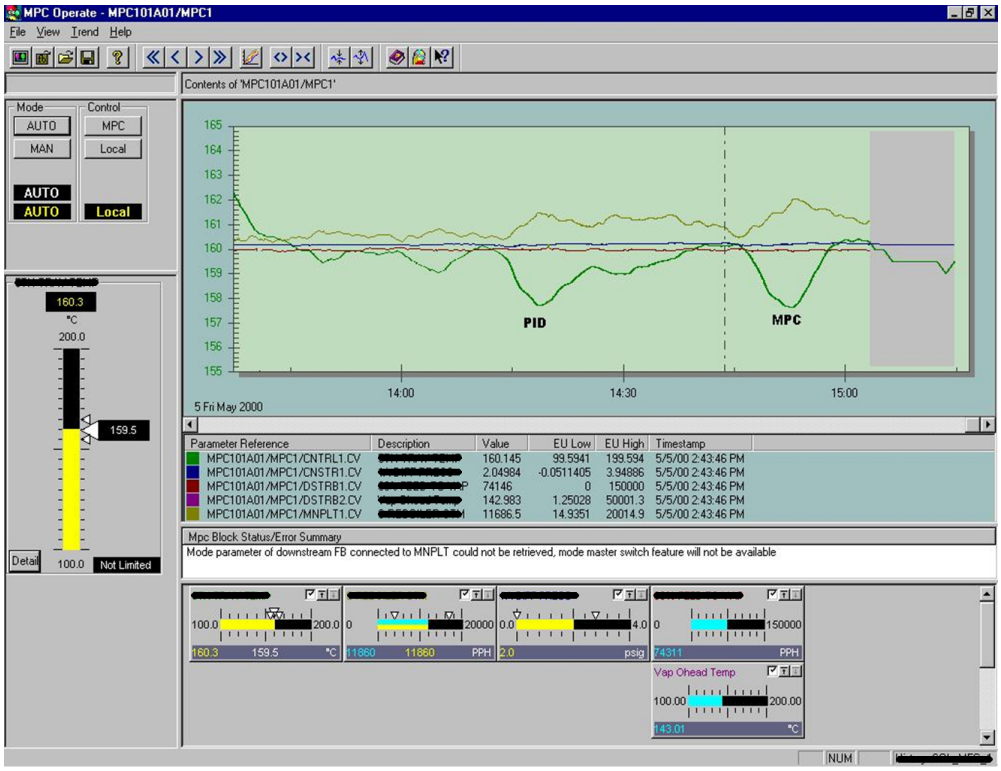

Figure 5.1: Screenshot of operator interface in chemical plant comparing PID and MPC disturbance rejection performance

When exposed to approximately the same unmeasured disturbance, MPC control shows a larger control error but brings the controlled variable back to the setpoint slightly earlier overall – depending on the specific production criteria, this may or may not be considered a small improvement. Therefore, much of this chapter is motivated by the question: why can't modern control algorithms, such as MPC, applied to practical plant applications, perform as well as PID, a technology developed more than 100 years ago? One of model predictive control's most prevalent and successful applications is, of course, multivariable control of distillation columns, which is, as with most multivariable applications, a process that is very difficult to control with PID

controllers. However MPC is often advertised and used for lag-dominant single loop applications, in which it usually performs worse than PID. This chapter presents a novel modification to the MPC feedback correction algorithm that can close the feedback performance gap between MPC and PID.

5.1 Drawbacks of model-based control

For setpoint tracking applications, the predictive controller's ability to store and update an approximation of the process state is the control performance advantage of model-based controllers over model free control. However, this same mechanism impairs the predictive controller's ability to move the controller output fast enough to react to an unmeasured disturbance scenario because the approximated state has to be corrected before the actual control error is calculated and any corrective control action can be taken. In other words, an unexpected output change has to cause a prediction update before it can cause a control move.

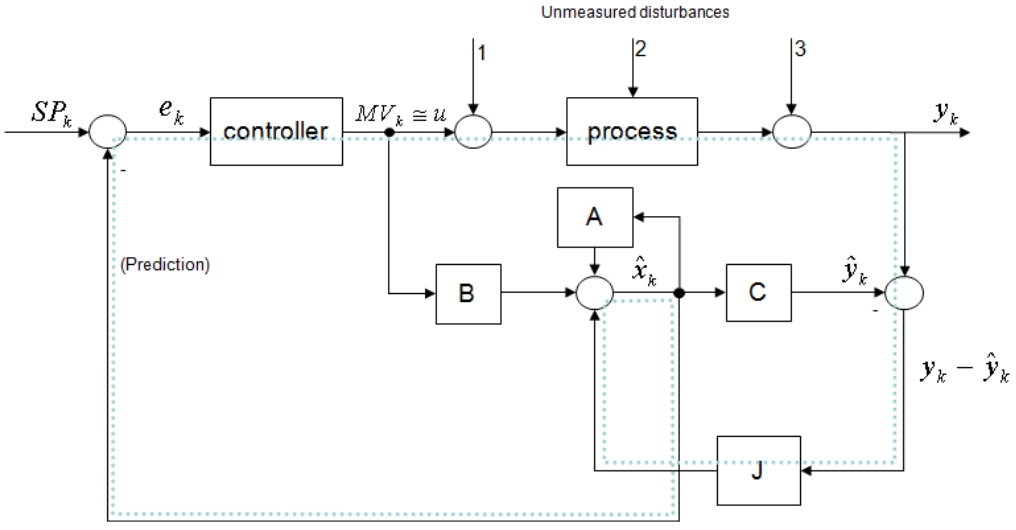

Figure 2.7: State update in closed-loop

The final closed loop control path, resulting from the insertion of a state estimator (J) into the controller, is shown in Figure 2.7 and was already discussed in section 2.3. In summary, it can be noted that, since the state observer is designed for a particular process model it will only result in optimal estimation and therefore optimal control in the absence of model mismatch. If the plant model drifts the observer itself may interfere with the controller's ability to react to a particular change in the controlled variable. The controller can only react to an unmeasured disturbance as fast as the disturbance impacts the prediction, which may be delayed by a mismatched state observer. This is why it is important to focus on the control performance rather than the state update performance when tuning an observer, as discussed more detailed in section 2.3. A model free controller, like PID, does not have this issue; the controller can immediately react to unmeasured disturbances. This chapter proposes a new method that allows the model predictive controller to react directly to unexpected changes in the controlled variable.

If it could be assumed that the model is perfect, the state update procedure is straightforward because the entire error term can immediately be attributed to unmeasured disturbances, i.e. full state update at each sample period. However all modern commercial MPC controllers have filter factors that even-out such changes over multiple sample periods to prevent instability because neither perfect nor linear models are expected in industrial applications. Such filter methods slow the control response down even further. Researchers have suggested methods that distinguish between unmeasured disturbance and model error [49]. Such methods, however, are very difficult and expensive to carry out during the runtime of a controller. Furthermore, even if it could be known exactly what fraction of prediction error

resulted from an unmeasured disturbance vs. model error, the corrective action of the controller would not be optimally fast until the model is re-identified.

The commissioning of model-based controllers natively involves matching the controller model parameters to the process model parameters. Model-based tuning methods for PID controllers are intended to simplify PID tuning by mimicking model-based controller behavior. Since the PID controller equation cannot account for deadtime, a Smith predictor setup allows matching the process deadtime to the Smith predictor deadtime. Methods like pole cancelation and lambda tuning match the integral time of the controller to the dominant time constant in the process. Lambda tuning adds an additional tuning parameter (λ), with which the engineer can pick a desired closed loop settling time.

As discussed in 1.1, model-based controllers excel in deadtime dominant single-loop processes because the deadtime model parameter can be matched to the equivalent deadtime controller parameter. Examples of such processes are flow loops, static mixers, and paper machines, while the most common examples of lag-dominant processes include temperature and pressure loops, and the chemical process at Solutia mentioned above. On the other hand Lag-dominant processes are better controlled by a tuned controller than a matched model-based controller or a PID controller with model-based tuning, which is clearly illustrated in Figure 1.2. The "break-even" point between tuned and matched controller is about θ/τ=0.3 for PI controllers and θ/τ=0.55 for PID controllers. In his work on controller tuning Shinskey [8] relates most of the reasons for this behavior to the integral action of a PID controller.

A model-based controller does not have any components that allow direct tuning of the integral action that is applied in the feedback calculation. While a linear

quadratic regulator (LQR) can be shown to have integral action when it is augmented with a Kalman filter for state update (2.8) - (2.9), the integral action is not independently tunable.

$$J = \int_0^{t_f} \left[xQx^T + uRu^T \right]$$

$$\dot{x}(t) = Ax(t) + Bu(t); \dot{z}(t) = -y(t)$$

$$\vdots$$

$$u = -K'x'(t) = -\left[K_x x(t) - K_z \int y(t)dt \right]$$

This combination is also known as the linear quadratic Gaussian (LQG) controller, a fundamental form of linear model-based control. As with PID controllers, integral action in model predictive controllers is necessary to control to the setpoint without leaving an offset.

From the above tuning discussion it is apparent that deadtime-dominant processes require a different set of controller features than lag-dominant processes. If too much delay exists, the best possible performance is $\frac{IE}{\Delta q K_q \tau_d}$=1, and a model-based controller with matched tuning is required. If too much lag is present, the performance can be better than $\frac{IE}{\Delta q K_q \tau_d}$=1 as long as it is *not* controlled by a model-based controller but by a PID with appropriate not matched integral tuning. The goal of the research discussed in this chapter is to create a controller that combines the advantages of both scenarios.

5.2 Tuning of model-based control

While the previous section discussed advantages and ideal operating ranges of model-based and PID controllers with respect to fractional deadtime, this section

compares the robustness of the two controller types. As mentioned above, model-based controllers, such as MPC, have better feedback performance in the deadtime dominant region of Figure 1.2. In the lag-dominant region, which is at least as common as the deadtime dominant region, the PID generally shows better feedback control performance. This has triggered many improvements to the original MPC algorithm, such as the addition of Kalman filtering, which have been widely accepted and implemented in industrial MPC products. Commercial model predictive controllers are now capable of better performance than traditional internal model control (IMC), as shown in Figure 1.2. While Chapter 3 addressed MPC tuning details, this chapter focuses on performance and tuning differences between model-based and PID controllers with respect to fractional deadtime. Figure 5.2a shows a comparison of three different MPC controllers for a first order plus deadtime process (FOPDT) with the following transfer function $G(s) = \frac{1}{50s+1}e^{-s}$. A PID with Skogestad tuning is included for comparison. Dynamic matrix control (DMC) originally used a prediction biasing calculation to account for prediction error. It achieves an integrated absolute error of 3.17, which for an unmeasured disturbance of $\Delta q = 1$ and the above transfer functions gives $\frac{IE}{\Delta q K_q \tau_d}$=3.17 > 1, presenting the worst feedback control performance of the compared controllers. If different implementations of Kalman filtering are used, the normalized integral error $\frac{IE}{\Delta q K_q \tau_d}$ becomes 0.68 and 0.35, respectively, representing a significant improvement over IMC with $\frac{IE}{\Delta q K_q \tau_d}$=1. In this scenario, however, MPC still falls slightly short of PID, which achieves $\frac{IE}{\Delta q K_q \tau_d}$=0.32. Such good performance numbers are only achieved because the balance between error and move term from

Equation (1.3)) is forced towards faster control moves in addition to the application of Kalman filtering. In other words, penalty tuning for controller speed becomes effective if the feedback path employs a Kalman filter. While penalty tuning has an effect on the DMC controller, it cannot bring the normalized integral error below 1. If the particular penalty tuning balance of Q=1 and R=0.1 from Figure 5.2a is further adjusted for performance to Q=1, R=0.01, then the normalized integral error to 0.21 and 0.14, respectively, is significantly improved, as shown in Figure 5.2b.

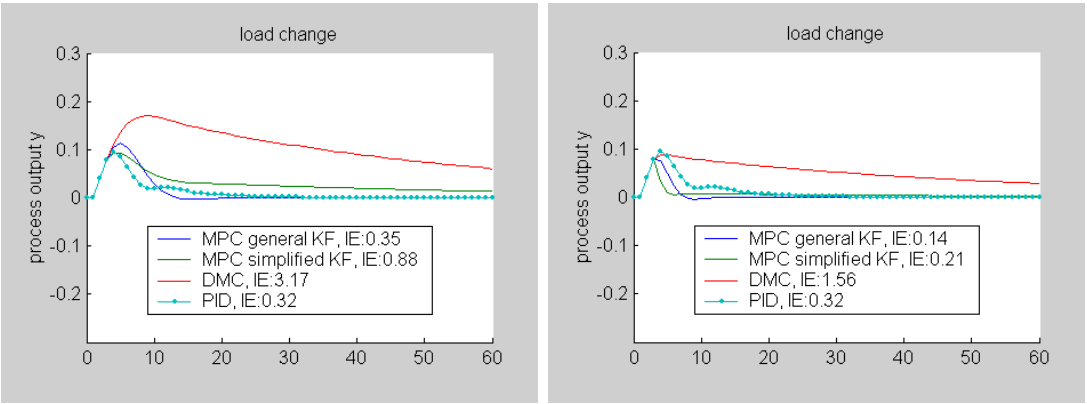

Figure 5.2: Disturbance rejection response K=1 T_1=50 T_2=0 θ=1, PID: (Skogestad) K_c=25 T_i=8 T_d=0, MPC: P=10 M=3 Q=1; on left a) R=0.1, on right b) R=0.01

This comparison shows that modern model predictive controllers can be tuned to perform better than classical internal model controllers based on PID control, i.e. better than $\frac{IE}{\Delta q K_q \tau_d}$=1 as long as Kalman filtering is used. The original purpose of adding tuning parameters to MPC, however, was to increase robustness in case of model mismatch by making the controller more sluggish. Using the same tuning to increase performance is clearly a negative use of the original idea and inevitably yields a loss of robustness, as seen in Figure 5.3. In this example, the model mismatch in the first order time constant is only 2 ($\tau/\tilde{\tau} = 2$) and model predictive control becomes

oscillatory. Figure 5.3 depicts the tuning tradeoff between performance and robustness in model predictive control with a Kalman filter. For the simplified Kalman filter implementation as described in Chapter 2, the integrated absolute error plot with the more balanced error and move terms of Q/R=10 (chart a) is notably flatter than that of the higher performance tuning with Q/R=100 (chart b); this indicates significantly more robustness to model mismatch in first order time constant. However chart b shows considerably better control performance around $\tau/\tilde{\tau} = 1$, i.e. if the model is known perfectly. What can be observed for general Kalman filter MPC with the same tuning if $\tau>\tilde{\tau}$ is the counterintuitive opposite of that of the simplified Kalman filter MPC. As the move penalties are reduced, the performance becomes better *and* the curve becomes flatter. This, however, does not hold true for $\tau<\tilde{\tau}$. This effect will be discussed in the next section, which considers higher order processes.

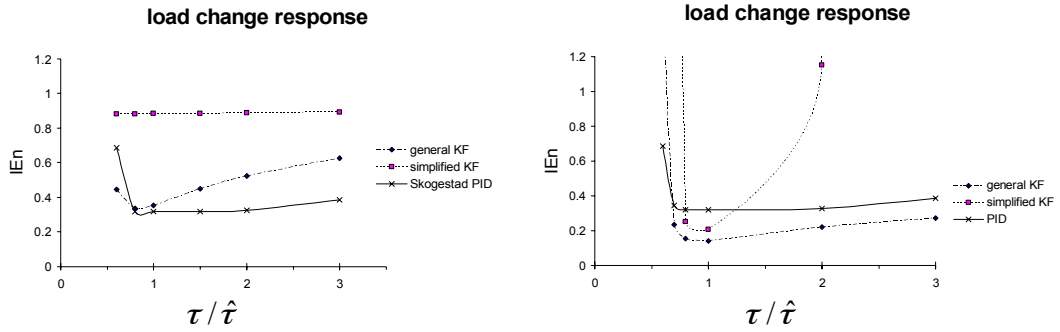

Figure 5.3: Feedback control performance depending on model mismatch and penalty tuning, K=1 T_1=50 T_2=0 θ=1, PID: (Skogestad) K_c=25 T_i=8 T_d=0, MPC: P=10 M=3 Q=1; on left a) R=0.1, on right b) R=0.01

Bearing the above in mind, it is obvious that MPC penalty tuning is arbitrary until one considers the impact of model mismatch. . Especially since the setpoint change performance is not worsened as the disturbance rejection performance is improved one could tent to unbalance the terms further and further in the direction of

the best performance. Chapter 3 discusses methods that automatically determine the optimal tradeoff based on different robustness and model mismatch criteria.

5.3 Tuning for industrial process characteristics

First order plus deadtime processes are not very representative of processes controlled by industrial control systems. Common processes in industrial plants include multiple physical, chemical or biological characteristics that change in series before the effect on a controlled variable is measured by a sensor. Industrial processes are often equivalent to a series of dynamic transfer functions. A common control loop includes, aside from the process, a number of valves, valve positioners and hardware sensors that may add many additional transfer functions to the loop, thereby potentially increasing the process model order significantly. Control device and transmitter manufacturers seek to reduce the impact that time constants of their products have on loop performance by using fast mechanical linkages and sensor materials. In spite of this, too much speed and too high sample rates conflict with noise reduction and aliasing objectives, and may also be very expensive and/or energy consuming. Shinskey points out the even though industrial processes often have many more time constants, the resulting overall curve shape can be approximated by a second order process model with little modeling error because the lags usually interact [8].

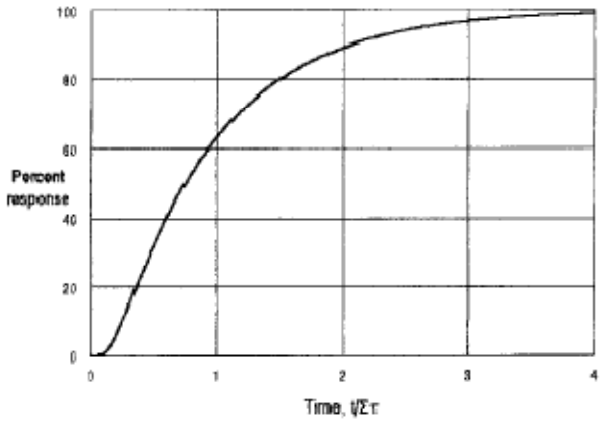

Figure 5.4: Step response of 20 interacting lags [8]

Figure 5.4 shows the step response of a distillation tower with 20 trays that effectively creates 20 independent lag time constants in series. The overall curve shape very closely resembles a second order curve and the resulting time constant may be calculated by (5.1).

$$\Sigma\tau = \tau_i \sum_{1}^{n} i = \tau_i \frac{n^2 + n}{2} \tag{5.1}$$

In industrial applications, first and second order approximation of a given controlled process are most common. Considering a certain amount of model mismatch, which can never be eliminated, second and third order process models are very comparable with respect to model error. Thus, industrial users usually model no more than two time constants, plus deadtime. This section discusses which of the previously discussed dynamic behaviors are different when controlling a second order plus deadtime (SOPDT) process. No significant differences could be found when comparing second to third order plus deadtime processes. Figure 5.5 shows how the tuning discussed in section 5.3 performs on both a FOPDT and a SOPDT process.

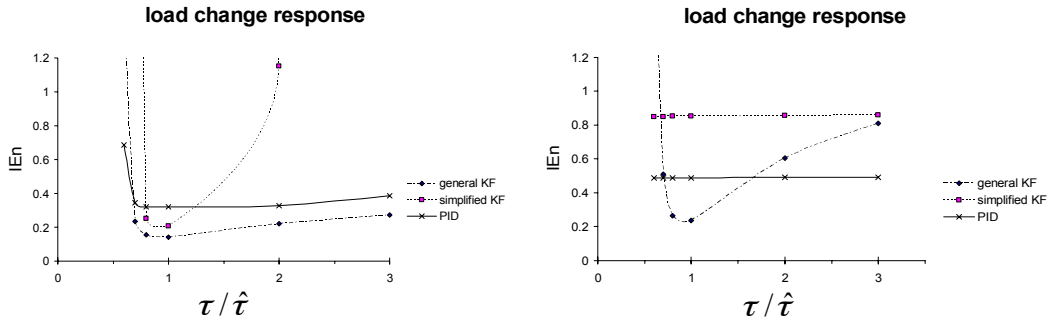

Figure 5.5: Feedback control performance depending on model mismatch and penalty tuning, K=1 T_1=50 T_2=0 θ=1, PID: (Skogestad) K_c=25 T_i=8 T_d=0, MPC: P=10 M=3 Q=1; on left a) R=0.1, on right b) R=0.01

As with the FOPDT, the SOPDT also achieves the best possible control performance at $\tau=\tilde{\tau}$ (with no model mismatch). However the IAE values are different since the two different controllers control two different processes: $G_{FOPDT}(s) = \frac{1}{50s+1}e^{-s}$ and $G_{SOPDT}(s) = \frac{1}{(30s+1)(20s+1)}e^{-s}$. Using the algorithm presented in Chapter 3, the optimal prediction and control horizons are also different between the two controllers. While P=10 and M=3 is ideal for FOPDT processes, the SOPDT process is best controlled with P=30 and M=9. This is not surprising, as the additional order adds another change in slope even though the total settling time of a FOPDT with $\tau = 50$ is very similar to that of a SOPDT with τ_1 =30 and τ_2 =20. The two Kalman filter formulations behave very different if they are connected to a first or second order process. While MPC with the simplified Kalman filter improves drastically throughout the entire range and the slope flattens significantly, MPC with general Kalman filter does considerably worse on the second order process. However, the control performance of MPC with general Kalman Filter seems to be much less sensitive to model error. Classical DMC is negatively affected by the introduction of the second order time constant because its prediction error correction algorithm, biasing the prediction vector, does not handle model mismatch very well. The presence of a second

order filter time constant works to the advantage of the model predictive controller with simplified Kalman filtering. The oscillations that occurred with fast tuning in the presence of model mismatch (Figure 5.6a) are completely dampened as shown in Figure 5.6b which plots the same penalty tuning and model mismatch for the MPC controlling the second order process.

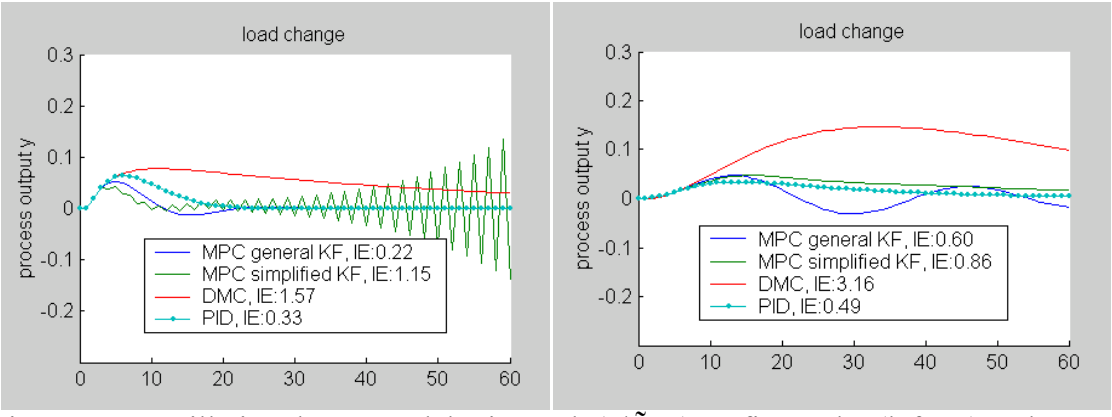

Figure 5.6: Oscillation due to model mismatch ($\tau/\tilde{\tau}$=2) on first order (left - a) and second order (right - b) processes, $G_{FOPDT}(s) = \frac{1}{50s+1}e^{-s}$ and $G_{SOPDT}(s) = \frac{1}{(30s+1)(20s+1)}e^{-s}$. PID: (Skogestad) K_c=25 T_i=8 T_d=0, MPC: Q=1 R=0.01; on left a) FOPDT, P=10 M=3 on right b) SOPDT P=30 M=9

As shown in Equation (2.10), simplified Kalman filtering is a state update method that simplifies tuning by using a tunable filter time constant in the feedback path. This time constant is set based on knowledge about signal-to-noise ratio.

$$G_{W_i}(q) = \frac{1}{q - a_i} \tag{2.10}$$

PID control is also much less sensitive to model mismatch in a second order process scenario than in a first order process scenario. This is fairly logical because a PID has two terms that can be used to compensate for two different process time constants. A PI controller is more suitable for controlling a first order process than

MPC with simplified Kalman filter. Given the above information, one could note that MPC with simplified Kalman filter shares more similarities with PID than MPC with general Kalman filter, in terms of tuning by means of filter time constant. This type of empirical tuning has a positive impact on performance if model mismatch is present.

The main conclusion that can be drawn from Figure 5.5 is that in all types of processes considered in this work, including higher order processes, the PID is outperformed by one of the MPC controllers, general or simplified, around $\tau \cong \tilde{\tau}$, when there is no or very small model mismatch. However due to the tuning that is required in a model predictive controller to achieve such control performance, the PID is much more stable in the presence of model mismatch; thus, PID outperforms MPC in process scenarios that are much more likely to occur as discussed in 1.1. The simple formulation of a PID controller allows the integral action to impact the outputs directly if an error is present for some amount of time. As discussed in 5.1, while this is advantageous for lead dominant processes, it is disadvantageous for deadtime dominant processes. The next section investigates a model-based controller with tunable integral action tuning as found in PID.

5.4 Augmenting tunable feedback to MPC

As discussed in the previous section, fractional deadtime plays a large role in determining whether PID-like integral action in MPC is appropriate or not and to what magnitude it should be tuned. The optimal tuning of integral action must, therefore, depend on fractional deadtime directly. While strong integral action greatly improves the feedback control performance on a lag-dominant process, it must be reduced or shut off completely in a deadtime-dominant process. In this section tuning of integral action

will be used to fit the MPC controller to the specific fractional deadtime properties of a process. The integral tuning may even be adapted online automatically if the process transitions between different regions of fractional deadtime. Such adaptation is practical because the tuning depends only slightly on model mismatch, as will be shown in this section. Tuning has much greater dependence on fractional deadtime, which is much easier to measure than model mismatch.

Since PI and PID are still the most popular feedback controller in the process control industry, the functionality and tuning of the I-term in a PID controller is well understood. In some cases, reset and other tuning parameters are calculated based on known tuning rules [19]. As mentioned before, some tuning rules require a process model, while others use closed loop characteristics, such as critical gain and critical period. Even if such parameters are unknown, plant operators and control engineers often intuitively know how to tune a controller incrementally from the current settings based on their experience. Figure 5.7 shows how the proportional and integral action of a PI controller impacts the controller's load performance for different process characteristics when an unmeasured unit step disturbance is introduced. Such charts make it easy to determine whether the ideal setting for a tuning parameter is above or below the current value. Using the knowledge represented by the charts, it is possible to find the ideal tuning by trial-and-error without the need to identify a process model.

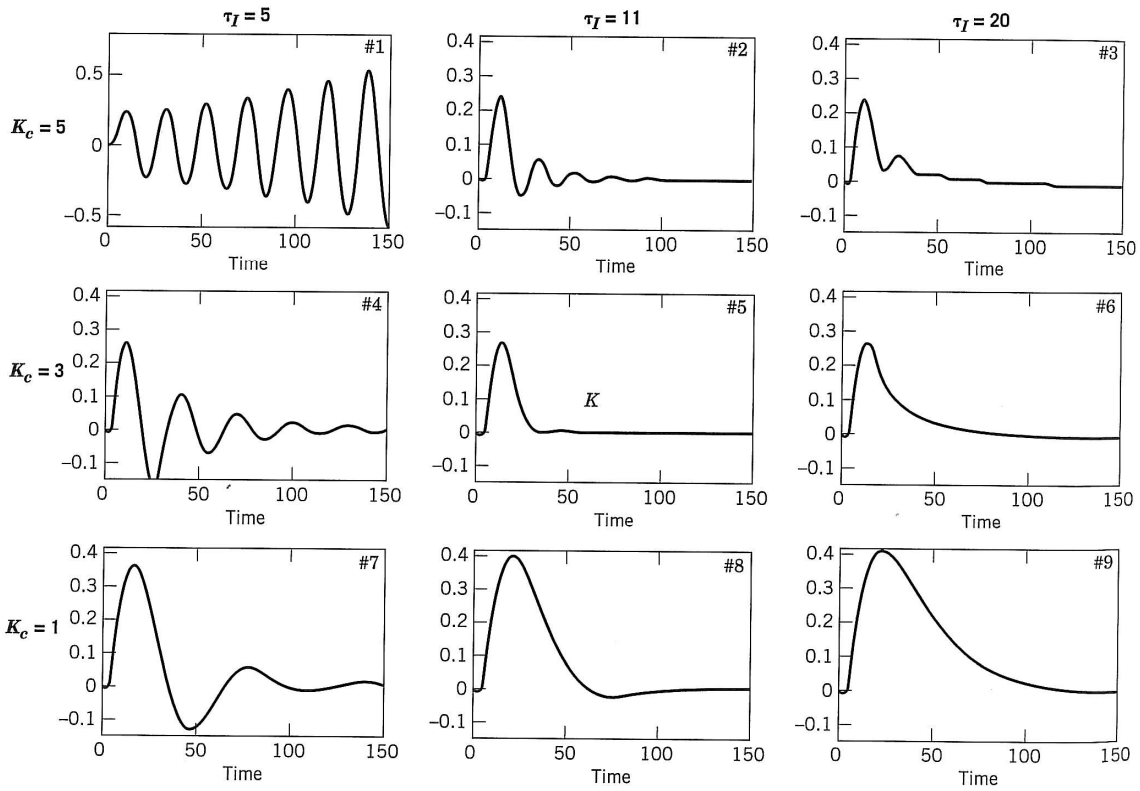

Figure 5.7: Unit step disturbance of PI controllers with different tuning settings. (FOPDT model: K=1, θ=4, τ=20). Ref [72]

Adding integral action (reducing T_i) speeds up the load disturbance rejection but usually impacts the setpoint change behavior negatively, as previously discussed. Section 1.4 discusses two degrees of freedom controller formulations that compensate for this problem, which applies to PID but not model-based control. Such tuning methods and control equation modifications are clearly not model-based, but empirical. Whenever researchers or practitioners add tuning parameters to the original three (gain, reset, rate), it is usually with the intent to better fit the PID algorithm to a particular usage scenario or group of usage scenarios. Such parameters simplify tuning for the specific application but often make it harder to tune based on model knowledge.

Sometimes they allow substituting of model knowledge with knowledge about actual vs. desired control performance.

As discussed in section 5.1, a model-based controller does not have any components that allow direct tuning of the integral action that is applied in the feedback calculation and the fact that model-based controllers can account for unmeasured disturbances without any offset means that they exhibit integral like action. However, it is not tunable. Figure 5.8 shows how tunable integral action can be augmented in the feedback path of a model predictive controller.

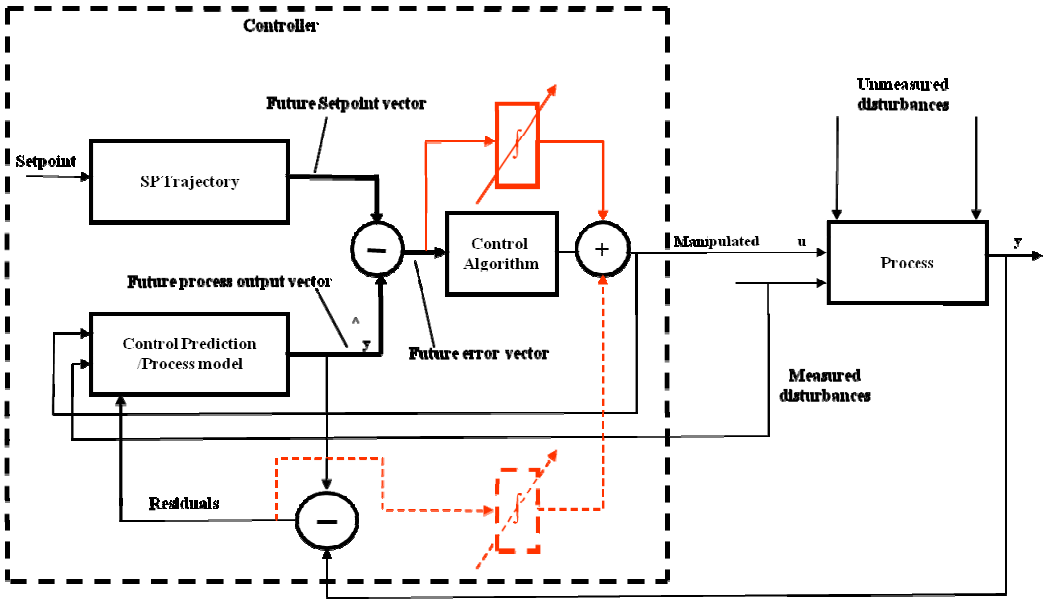

Figure 5.8: Tunable integral action in the feedback path of a model predictive controller

The augmented integrator can be used to integrate the residuals (dotted line) or the future error vector (solid line). The latter generally achieves the best results because the residuals are only one of the contributing factors to the error vector, and thus have less impact on the control action calculated by the control algorithm. Factors that can

reduce the impact of a direct integral action which is augmented to residuals are filter methods for residuals and the setpoint. In both scenarios, the characteristic PID integral behavior is achieved by adding to the controller output directly, as done in a PID controller. This allows the controller to react to an unmeasured disturbance more rapidly and within the same sample period as the disturbance occurred. The huge benefit of a model predictive controller is that it is able to pre-calculate multiple future moves based on the process model and future prediction of process outputs.

$$\min_{u^N} \sum_{j=0}^{\infty} (y_{k+j}^T Q y_{k+j} + u_{k+j}^T R u_{k+j} + \Delta u_{k+j}^T S \Delta u_{k+j}) \tag{1.3}$$

When MPC as defined in (1.3) is implemented in a time discrete sampling control system, it results in a recursive algorithm that uses filter-time constants to correct for unmeasured disturbances and model mismatch: While the predicted future-process outputs and target setpoint trajectory are used to calculate the future-control moves, the predicted future moves are in turn used to update the state variables, i.e., the future output prediction. A standard MPC as described in (1.3) with default tuning will correct an unmeasured disturbance as fast as possible in an optimal sense. However, if there is model mismatch, this recursive calculation will be unbalanced for a longer time than without model mismatch, which is what the integral term of a PID controller is designed to pick up and act on. It is therefore possible to distinguish which part of the future error vector, as shown in Figure 5.8, is a result of setpoint change, versus unmeasured disturbance. Depending on the tuning of the integral action, the actual effect of integral tuning will only be noticeable if model mismatch is present. In the absence of model mismatch, less integral action may be required for the best control performance. As can be seen in Figure 5.10, for MPC with general Kalman filter the

integral action adds very little benefit; at t/t=1 the performance-increasing effect of integral action becomes more and more obvious as the model mismatch increases. The resulting control performance charted over time—adding tunable integral action to the three types of model predictive controller—is depicted in Figure 5.9.

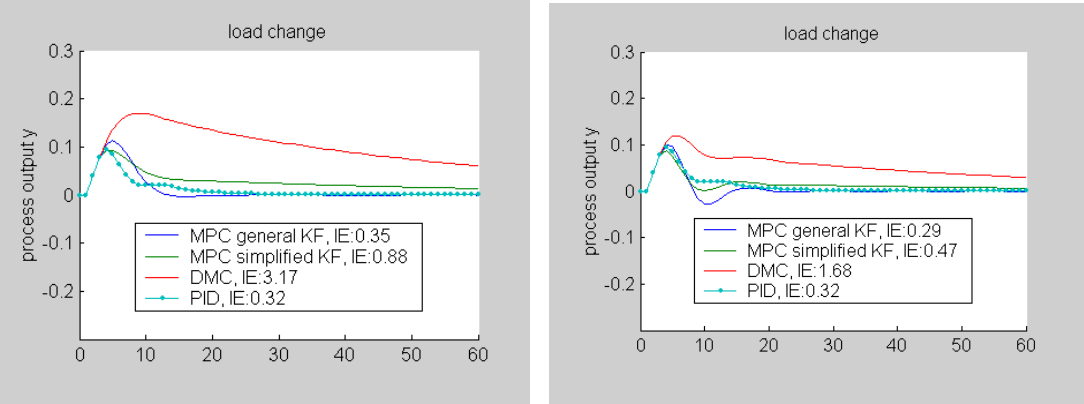

Figure 5.9: Comparison of load rejection performance before and after adding tunable integral action to the future error vector calculation

The PID response is plotted for reference. All three MPCs show a significant performance improvement. The greatest absolute improvement can be found on the DMC controller. Its IAE is reduced by 1.49 which is a 47% improvement from the standard DMC formulation. MPCs with general and with simplified Kalman filter improve by 29.1% and 46.6%, respectively. The most remarkable finding, however, is that as the performance improves the robustness does not suffer at all. Figure 5.10 shows a comparison of performance in relation to model mismatch. The MPC plots in Figure 5.10b not only dropped as compared to Figure 5.10, but also becomes flatter than, or as flat as, those without augmented integral action.

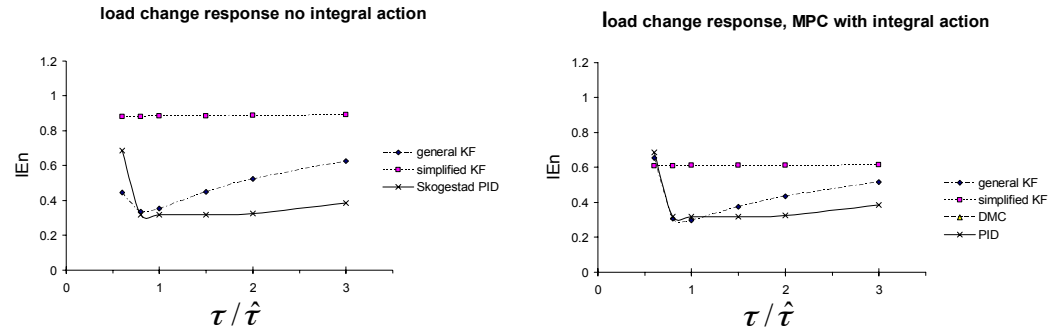

Figure 5.10: Comparison of robustness before and after adding tunable integral action to the future error vector.

The tuning of the integral action is calculated by Skogestad rules, the same way a PID controller would be tuned, leading to an integral time of T_i=8s. Manual tuning of the augmented integrator can further improve the feedback control performance. The empirical tuning described in Figure 5.7 resulted in a stronger integral action with T_i=4s, shown in Figure 5.11. The integral action can actually be increased even further until, at Ti=2, the curve slope becomes steeper again, indicating a drip in robustness without a noticeable gain in performance.

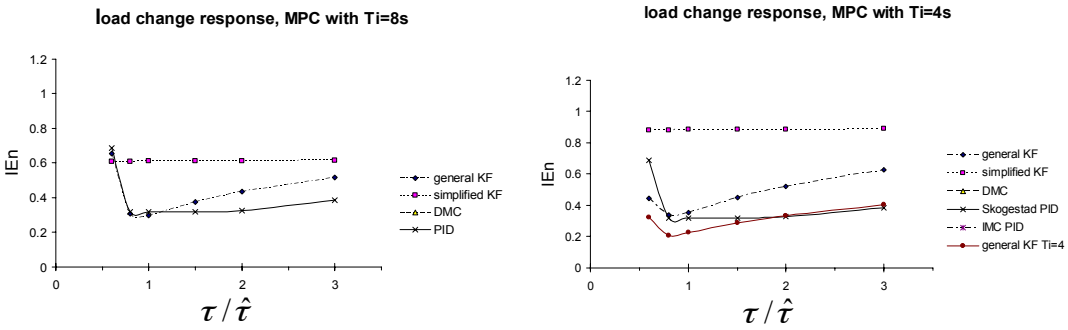

Figure 5.11: Comparison of robustness and performance with manual tuning of integral action on the future error vector

The same tuning analysis was run on the aforementioned second order process and model predictive controller tuning. As shown in Figure 5.12, the outcome is surprisingly different. While augmenting an integrator to a FOPDT loop benefitted all

model predictive controllers, augmenting the exact same integrator to a SOPDT loop only improved the performance of the MPC with a simplified Kalman filter. As mentioned above, this type of MPC is relatively insensitive to model mismatch when applied to a second order process. Thus, lowering the nearly flat performance plot of the MPC with simplified Kalman filter by such a significant amount results in overlap with the curve of MPC with general Kalman filter at about $\tau/\tilde{\tau} = 2$.

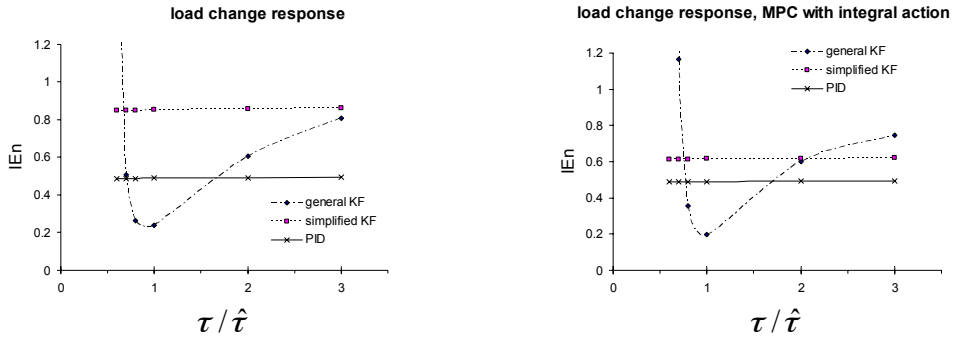

Figure 5.12: Comparison of robustness and performance with manual tuning of integral action on the future error vector on second order process

Therefore, we can see that while it is advantageous to use general Kalman filtering on a second order process, if no or very little model mismatch is expected, one should switch to simplified Kalman filtering if model mismatch is expected to be outside of $0.75<\tau/\tilde{\tau}<2$.

If setpoint changes are introduced to a model predictive controller with augmented integral action, one can observe the same negative effect on setpoint change performance as seen on PID controllers. Similarity, as with PID control, one can use a two degrees of freedom formulation to reduce or completely cancel out this effect.

5.5 Summary

This chapter showed that the control performance of PID is less impacted by model mismatch than model-based controllers, such as MPC. Furthermore, the feedback control performance, i.e. the rejection of unmeasured disturbances, of PID controllers is better than that of MPC controllers to start with as long as a process is lagtime-dominant. If the deadtime fraction changes to deadtime-dominant, then MPC is better at controlling the process because it requires an internal model to account for deadtime. A method was presented that combines the features of MPC and PID, thereby positively impacting the feedback control performance. This method takes advantage of model-based control and integral tuning in the appropriate region of fractional deadtime. While MPC has inherent integral action that allows offset-free control, the tuning of integral action is critical to feedback control performance. Conventional PID tuning rules that favor feedback control performance or adaptive tuning can be applied to calculate reasonable tuning. Since the integral action is applied to the future error vector calculation only, it automatically becomes more prominent if the error is caused by model mismatch (as opposed to disturbance or setpoint change). In other words, while the integral action contributes when it is needed, it does not diminish performance when it is not needed. Augmented integral action improves control performance without reducing robustness.

Another interesting finding was that the impact of integral action drastically differed if a first or second order process was being controlled. If an algorithm, as the one presented in Chapter 3 is used to automatically determine all tuning parameters through optimization, then the model order is an important variable in the objective

function. However, significant differences are only observed between first and second order processes. The difference between second and higher orders is negligible.

While the presented method improves the feedback control performance of model predictive control significantly, PID still performs better than MPC if model mismatch is present, i.e. for most of the values for $\tau/\tilde{\tau} \neq 1$ in Figure 5.10 to Figure 5.12. This indicates that there is another feature of PID controllers that is beneficial during model mismatch and that might be added to augment MPC. Further research could investigate the benefits and challenges of augmenting derivative action to MPC, in addition to integral action discussed here.

Based on the newly developed technology suggested in this chapter and the demonstrated feedback control performance improvements the author is confident that this method would have had a great positive impact on a challenging process as the one from Solutia described in the beginning of this chapter. It is conceivable that the MPC control performance would have been as good or better than the PID control performance discussed in the beginning of this chapter.

Chapter 6

Conclusions and Recommendations

Despite many attempts to broaden the range of applications for model predictive controllers, their use is still restricted mainly to feedforward, decoupling, constrained control and optimization applications. One of the reasons for this can be found in the types of processes than are not controlled well by advanced control strategies such as MPC. This research focused on identifying these types of processes and analyzing deficiencies of model predictive controllers for these particular processes. Usually, when MPC is applied there are high expectations for control performance improvement. Users of this technology are often unaware of its limitations when applying it. Since in model predictive control strategies the control performance is largely dependent on the precision of the plant model, as was shown in Chapter 2, control performance can degrade very rapidly in some tuning scenarios if model mismatch is present. Specifically, the feedback control performance is negatively affected by model mismatch.

The objective of this research was to improve MPC control performance in scenarios that are currently predominantly solved with PID controllers. The intention was not to create another PID replacement controller, as has been attempted numerous times in the past with algorithms like Fuzzy Logic, MPC and others. The intention was to be able to use the advanced features of MPC in more process scenarios, specifically feedback control scenarios, where MPC had previously been unable to perform as well as PID. In other words, the goal of this research was to make MPC applicable to more types of processes.

6.1 Summary of contributions

Chapter 2 analyzed the impact of model mismatch to control performance. There is abundant literature on state update, and many methods have been developed that improve the performance of the state update component of a model predictive controller. The problem of model mismatch is well known in industry and academia. Since model based controller performance can be severely degraded by model mismatch, the majority of efforts to improve control performance in model predictive controllers is focused on improving the performance and precision of state update algorithms that correct for the effects of model mismatch. Another large portion of control literature is concerned with improving initial model quality, adapting process models to process parameter changes, or detecting process parameter changes in order to inform the user and trigger a manual process model reevaluation. The research described in this dissertation investigates the direct control performance impact of state update mechanisms, and shows that optimal control performance can be achieved with suboptimal state update performance (and vice versa). To date, almost no literature can be found in this area. Additionally, a very detailed analysis of the impact of controller tuning in the presence of model mismatch was performed and documented during the course of research. The tuning relations that are investigated in this work not only include well known penalty weights, but also some that are not traditionally considered tuning parameters. Examples include modeling and control horizons, Kalman filter constants, and types.

Many previously developed solutions that improve MPC control performance in realistic process scenarios, such as disturbance modeling [20], require knowledge of a disturbance model and its parameters. Most of such modern methods are derived for

perfect models, and lose optimality if model mismatch occurs. Chapter 3 of this dissertation presented a new automatic method that optimizes control performance for a given model error by controller tuning, rather than by model improvement or state update modification. One underlying challenge throughout this research was that no assumption could be made regarding the process model quality, since in a real plant there is no such thing as a perfectly known model. This precondition was motivated by the author's plant experience during his role in the development of commercial MPC and control system software. The novel method presented in this work provides optimal tuning not only for certain fixed amounts of model mismatch, but also for a range of model mismatch because both the presence and the change in model mismatch (is usually to be expected. It can do so for any preselected range of 'worst case' model mismatch because an optimization is used to find the best location of an assumed controller model sub space within the space of optimal tuning values.

Chapter 4 explained the application of this new method and discussed how it can be used to adapt controller tuning in the following four scenarios (and various combinations thereof):

1. Manual entry of the actual model, if known
2. Automatic property estimation or model identification based on plant inputs and outputs
3. Manual entry of model mismatch ranges, if known
4. Automatic estimation of model mismatch based on innovation analysis from state estimation

Finally, this chapter proposes an adaptive method using existing technology (autocorrelation of innovation).

While the other chapters introduce methods that compensate for known and unknown model mismatch in an optimal way, the unique contribution of the method described in Chapter 5 is that the feedback control performance of a model predictive controller can be improved by a simple addition of tunable integral action. The tuning of such integral action can be done easily by using the well known and well tested tuning rules that are used for PI and PID tuning. One big advantage of PID is that it can be tuned without knowledge of process models. PID tuning methods have been developed that utilize simple factors to configure desired gain and phase margins [22] to accommodate a user-specifiable tradeoff between performance and robustness. Many tuning methods allow testing the process with very little perturbation [23]. At the same time, the abundance of tuning rules can be a disadvantage [19]. Chapter 5 proposed a novel method to improve model predictive control performance by augmenting tunable integral action. The contribution to the field is a method that uniquely utilizes knowledge of the dynamic behavior of disturbances to calculate corrective action that can be added directly to the manipulated variable of a model predictive controller. This is similar to how a PID algorithm includes past error characteristics in the calculation of corrective action and improves the feedback control performance dramatically for control scenarios with model mismatch and/or changing dynamics. No such algorithm is used in today's MPC controllers. Future research may investigate stability margins and an automatic method to find the optimal tuning parameters. The impact of augmenting derivative action might also be investigated in future research.

6.2 Recommendations for future research

There are several courses that future research might take, such as investigating more criteria, like the effect of deadtime, and more complex process models or multivariable processes. These further degrees of freedom were consciously omitted in this work, however, for logical or scope reasons. The intention of this research was to present new concepts for improving control performance in the presence of model mismatch. These concepts were tested in simulation and in plant experiments that involved an actual digital control system which was connected to real plant equipment and running a chemical process. Further research might also investigate more additional criteria, such as additional tuning parameters, higher order models, or more customized definitions of control performance and stability.

Additionally, selected parts of this research could be extended to multivariable scenarios, as large process handling and optimization are two of the key selling features that MPC has over other control strategies, such as PID.

The conceptual design of a fully automatic adaptive controller that uses the optimal MPC tuning method that has been presented in Chapter 4 is far from being complete. Further research in this area should investigate how multiple alternate tuning scenarios can be implemented in an automatic fashion with minimal impact to plant operation, and without affecting the autocorrelation method itself or creating positive feedback. Further research in this area could investigate the ideal model update trigger functions or even gain scheduling methods. Before commercial usage or productization of such automatic method is feasible, more sophisticated safety nets have to be developed. A wider variety of process and model types may be researched in future

research. Also work could be done to quantify and normalize the amount of model mismatch as an easy to understand generic metric.

Just as Chapter 5 discussed how the feature of tunable integral action can improve MPC control performance, future research could investigate the use of tunable derivative action. Derivative tuning is a well known and understood feature in the user community that is often used to overcome the effects of unknown nonlinearities and has potential to benefit modern advanced control strategies such as MPC.

Bibliography

[1] C. R. Cutler and B. L. Ramaker, Dynamic matrix control – a computer control algorithm, Proc. Automatic Control Conf. Paper WP5-B, San Francisco, CA, 1980

[2] R. E. Otto, Forward modeling controllers: a comprehensive SISO controller, AIChe meeting, November 5, 1986

[3] G. Pannocchia, N. Laachi, J. B. Rawlings, A candidate to replace PID Control: SISO Constraint LQ Control, DYCOPS Proceedings, (2004)

[4] ADCO – adaptive Riccati controller, 1992, http://www.ipas-systeme.de/products111.htm

[5] A. Hugo, Limitations of model predictive controllers, Hydrocarbon Processing, Jan 2000, pp 83-88

[6] M. Jelali. An overview of control performance assessment technology and industrial applications. Control Engineering Practice, 14:441–466, 2006.

[7] T. Olsen, B. Bialkowski, EnTech Control, Session No. 42 - Applications of Control to Refining, AIChE Spring National Meeting in New Orleans, March 2002

[8] G. Shinskey, Process Control: As Taught vs. as Practiced, Ind. Eng. Chem. Res. 41, 3745-3750, 2002

[9] S. Skogestad, Simple analyticRules for Model Reduction and PID Controller Tuning, J. Process Control 13, 291, 2003

[10] K. Aström, T. Hägglund, PID Controllers Theory, Design and Tuning, 2nd ed., ISA, Research Triangle Park, NC, 1995

[11] T. L. Blevins, G. K. McMillan, W. K. Wojsznis and M. W. Brown, Advanced Control Unleashed, ISA, 2003, ISBN 1-55617-815-8

[12] A. S. Morse, F. M. Pait, S. R. Weller, "Logic-Based Switching Strategies for Self-Adjusting Control," 33rd IEEE Conference on Decision and Control, Workshop Number 5. Lake Buena Vista, Florida, USA, December 1994.

[13] K. S. Narendra, and J. Balakrishnan, "Adaptive Control Using Multiple Models," IEEE Transactions on Automatic Control,. Vol. 42, No. 2, pp.177-187, February 1997.

[14] G. K. McMillan, Good Tuning: a Pocket Guide, 2nd edition, ISA, 2005, ISBN 1-55617-940-5

[15] D. Thiele, T. L. Blevins, W. K. Wojsznis, Autotuning In Distributed Environment, ISA1999, Philadelphia

[16] D. Thiele, W. K. Wojsznis, A. Mehta, Achieving nonlinear MPC performance with neural network aided state update, ISA 2004, Houston

[17] D. Thiele, Benefits and challenges of implementing model predictive control as a function block, ISA2000, New Orleans

[18] Integrated model predictive control and optimization within a process control system, US Pat. 7050863 - Filed Jul 25, 2003 - Fisher-Rosemount Systems, Inc.

[19] A. O'Dwyer, PI and PID controller tuning rules for time delay processes: a summary. Dublin Institute of Technology, Dublin, Ireland, 15 May 2000

[20] T. Badgwell, K. Muske, Disturbance Model Design for linear Models. Proceedings of ACC, 1621-1626, Anchorage, AK, May 2002

[21] P. Lundstrom, J. Lee, M. Morari, S. Skogestad, Limitations of dynamic matrix control. Computers in Chemical Engineering 19, 409-421, 1995

[22] W. Wojsznis, J. Gudaz, T. Blevins and A. Mehta, Practical approach to tuning MPC, ISA transactions (ISA trans.), 42(1), 149–162, 2003

[23] W. Wojsznis, T. Blevins: „Systems and Method for automatically tuning a Process Controller", US Patend Number 5,453,925, September 1995

[24] G. Shinskey, Feedback Controllers for the Process Industries, McGraw Hill, 1994

[25] G. Shinskey, PID-deadtime control of distributed processes, 2001

[26] R.E. Kalman, A new approach to linear filtering and prediction problems, Journal of Basic Engineering 82 (1): 35–45, 1960

[27] M. Morari, N. L. Ricker, Model Predictive Control Toolbox User's Guide, 1998

[28] G. McMillan, Implementing MPC to Reduce Variability by Optimizing Control Valve Response, 2005

[29] R. Cagienard, P. Grieder, E. Kerrigany and M. Morari, Move Blocking Strategies in Receding Horizon Control, 43rd IEEE CDC, 14-17 Dec. 2004, Page(s): 2023 - 2028 Vol.2

[30] Y. Takahashi, M. J. Rabins, D. M. Auslander, Control and Dynamic Systems, Addison-Wesley, Reading, Massachusetts (1972)

[31] J. H. Lee, M. Morari and C. E. Garcia, State Space interpretation of model predictive control, Automatica, Vol 30, No. 4, pp. 707-717 (1994)

[32] S. Qin, T. Badgwell, A Survey of Industrial Model Predictive Control Technology, Control Engineering Practice, Volume 11, Issue 7, July 2003, Pages 733-764

[33] Bo Han; Xinggang Lin, Adapt the steady-state Kalman gain using the normalized autocorrelation of innovations, Signal Processing Letters, IEEE, Volume 12, Issue 11, Nov. 2005 Page(s): 780 – 783

[34] J. Maršik and V. Strejc, Application of Identification-free Algorithms for Adaptive Control, Automatica, Vol. 25, No. 2, pp.273--277, 1989.

[35] J. Maršik, P. Jaroslav, A nonlinear adaptive PID controller, IFIP Conference on Optimized-Based Computer-Aided Modeling and Design, Prague, Czech Republic, May, 1994.

[36] D. Thiele, W. Wojsznis, Multi Model Adaptive Industrial MPC Controller, Proceedings Control and Applications, Montreal, Canada, 2007

[37] W. Wojsznis, T. Blevins, Evaluating PID adaptive techniques for industrial implementation, Proceedings of the American Control Conference 2002, Page(s): 1151 - 1155 Volume 2

[38] P. Kesavan and J. H. Lee, A set based approach to detection and isolation of faults in multivariable systems, Elsevier Science Ireland, 2001

[39] B. J. Odelsen & Rawlings, Estimating Disturbance Covariances From Data For Improved Control Performance, Dissertation University of Wisconsin - Madison, 2003

[40] F. G. Shinskey, PID-deadtime control of distributed processes, Control Engineering Practice, Volume 9, Number 11, November 2001 , pp. 1177-1183(7), 2001

[41] G. Pannocchia, N. Laachi, and J. B. Rawlings. A candidate to replace PID control: SISO constrained LQ control. AIChE J., 51(4):1171-1189, 2005

[42] T. Hägglund, K. Aström, Method and an apparatus in tuning a PID-regulator - US Patent 4,549,123, 1985

[43] DeltaV, digital Process automation system, www.EasyDeltaV.com

[44] G. Pannocchia, G. and E. C. Kerrigan, Offset-free Control of Constrained Linear Discrete-time Systems Subject to Persistent unmeasured disturbances, In 42nd IEEE conference on decision and control, Hawaii, USA (2003)

[45] W. K. Wojsznis, T. L. Blevins, D. Thiele, Neural Network Assisted Control Loop Tuner, 2005, patent number 6847954

[46] CJCSim; http://www.cutlerjohnston.com/products.htm

[47] http://www.aspentec.com/includes/product.cfm?IndustryID=0&ProductID=54

[48] http://www.hyprotech.com/products/family.asp?ID=1

[49] A. Badwe, R. Patwardhan, S. Patwardhan, R. Gudi, Quantifying the Impact of Model-Plant Mismatch on Controller Performance: A non-invasive approach, International Symposium on Advanced Control of Industrial Processes (ADCONIP 2008)

[50] C. Desoer, M. Vidiyasagar, Feedback Systems: Input-output properties, Academic Press, New York, 1975

[51] N. Stanfelj, T. Marlin, J. MacGregor, Monitoring and Diagnosis of Process Control Performance: The Single Loop Case, Ind. Eng. Chem. Res., 32, 301-314, 1993

[52] P. Kesevan and J.H. Lee. Diagnostic tools for multivariable model-based control systems. Ind. Eng. Chem. Res., 36(7):2725–2738, 1997

[53] B. J. Odelson and J.B. Rawlings. Online monitoring of MPC disturbance models using closed-loop data. In Proceedings of the American Control Conference, pages 2714–2719, Denver, CO, 2003.

[54] J. M. Maciejowski, Predictive Control with Constraints, Prentice Hall, 2002, Shell oil fractionator, p 248

[55] OPC Foundation: http://www.opcfoundation.org

[56] DeltaV Tune: http://www.easydeltav.com/pd/PDS_DeltaV_Tune.pdf

[57] Emerson Process Management, DeltaV Documentation

[58] Kenneth R. Muske and James B. Rawlings. Model predictive control with linear models. AIChE J., 39(2):262-287, 1993.

[59] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. Automatica, 14:413-428, 1978.

[60] C. R. Cutler and B. L. Ramaker. Dynamic matrix control|a computer control algorithm. In Proceedings of the Joint Automatic Control Conference, 1980.

[61] D. M. Prett and R. D. Gillette. Optimization and constrained multivariable control of a catalytic cracking unit. In Proceedings of the Joint Automatic Control Conference, 1980.

[62] C. E. Garcia and A. M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). Chem. Eng. Commun., 46:73-87, 1986.

[63] P. Grosdidier, B. Froisy, and M. Hammann. The IDCOM-M controller. In T. J. McAvoy, Y. Arkun, and E. Zafiriou, editors, Proceedings of the 1988 IFAC Workshop on Model-based Process Control, pages 31-36, Oxford, 1988. Pergamon Press.

[64] P. Marquis and J. P. Broustail. SMOC, a bridge between state space and model predictive controllers: Application to the automation of a hydrotreating unit. In T. J. McAvoy, Y. Arkun, and E. Zafiriou, editors, Proceedings of the 1988 IFAC Workshop on Model-based Process Control, pages 37-43, Oxford, 1988. Pergamon Press.

[65] R. Shridar, D. Cooper, A tuning strategy for unconstrained multivariable model predictive control, Ind. Eng. Chem. Res., 37(10), 4003–4016, 1998

[66] J. Trierweiler, L. Farinab, RPN tuning strategy for model predictive control, J. of Process Control, 13(7), 591–598, 2003

[67] J. Garriga, M. Soroush, Eigenvalue Pole Placement Using Model Predictive Control, 2007

[68] S. Patwardhan, S. Shah, From data to diagnosis and control using generalized orthonormal basis filters. Part I: Development of state observers, J Process Control 15, 2005

[69] T. S. Schei, On-line Estimation of process control and Optimization Applications, 8th International IFAC Symposium on Dynamics and Control of Process Systems, Preprints Vol. 2, June 6-8, Cancún, Mexico

[70] B. Han, X. Lin, Adapt the Steady-State Kalman Gain Using the Normalized Autocorrelation of Innovations, IEEE Signal Processing Letters, Vol. 12, No. 11, November 2005

[71] S. Ko, R. R. Bitmead, State Estimation of Linear Systems with State Equality Constraints, IFAC 2005

[72] D. E. Seborg, T.F. Edgar, and D.A. Mellichamp, Process Dynamics and Control 2nd Ed.,New York: Wiley, 2004

Vita

Dirk Thiele was born in Magdeburg, Germany on December 25th 1973, the son of Arthur Gustav Thiele and Monika Marion Thiele. He received his Bachelor of Science Degree in Electrical Engineering from the polytechnic University of Magdeburg in 1998 while he was working at the Institut für Automation und Kommunikation Magdeburg (ifak), a non-profit institute for applied research in process control, automation technology and industrial communication. The same year he started his employment at Fisher-Rosemount Inc., later Emerson Process Management, in Austin, Texas. In the different roles that he has held over the last 11 years, including software developer, software designer and software engineering manager, he has been involved in the research, development and implementation of advanced control products for plant-wide performance monitoring, adaptive control, neural networks, model predictive control and optimization as part of the automation system DeltaV.

In August of 2001, he entered the Graduate School at the University of Texas at Austin where he was admitted to Ph.D. candidacy in January 2004 and began research under the guidance of Professors Robert H. Flake in the Department of Electrical and Computer Engineering and Thomas F. Edgar in the Department of Chemical Engineering.

Permanent address: 7811 Lonesome Dove
Austin, Texas 78729